AFATL-TR-88-155

Image Algebra Synthesis

G X Ritter
Joseph N Wilson
Jennifer L Davidson

DEPARTMENT OF COMPUTER AND
INFORMATION SERVICES
UNIVERSITY OF FLORIDA
GAINESVILLE, FL 32611

MARCH 1989

FINAL REPORT FOR PERIOD JANUARY - DECEMBER 1987

AIR FORCE ARMAMENT LABORATORY
Air Force Systems Command ▮ United States Air Force ▮ Eglin Air Force Base, Florida

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The AFATL STINFO program manager has reviewed this report, and it is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

Nicholas C. Hablenko

NICHOLAS C. HABLENKO, Lt Col, USAF
Acting Chief, Advanced Guidance Division

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization, please notify AFATL/AGS, Eglin AFB FL 32542-5434.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFATL-TR-88-155
6a. NAME OF PERFORMING ORGANIZATION University of Florida 6b. OFFICE SYMBOL (If applicable) CIS	7a. NAME OF MONITORING ORGANIZATION Air-to-Surface Guidance Branch Advanced Guidance Division
6c. ADDRESS (City, State, and ZIP Code) 301 CSE University of Florida Gainesville, Florida 32611	7b. ADDRESS (City, State, and ZIP Code) Air Force Armament Laboratory Eglin Air Force Base, Florida 32542-5434
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFATL/AGS & DARPA/TTO 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F08635-84-C-0295
8c. ADDRESS (City, State, and ZIP Code) AFATL/AGS, Eglin AFB, FL 32542-5434 DARPA/TTO, 1400 Wilson Blvd, Arlington, VA	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 62602F; PROJECT NO. 2068; TASK NO 06; WORK UNIT ACCESSION NO. 44
11. TITLE (Include Security Classification) Image Algebra Synthesis	
12. PERSONAL AUTHOR(S) G. X. Ritter, Joseph N. Wilson, Jennifer L. Davidson	
13a. TYPE OF REPORT Final	13b. TIME COVERED FROM Jan 87 TO Dec 87
14. DATE OF REPORT (Year, Month, Day) March 1989	15. PAGE COUNT 58
16. SUPPLEMENTARY NOTATION Availability of this report is specified on verso of front cover.	

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP
16	04	
17	07	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Image Algebra
Mathematical Morphology
Image Processing
Pattern Recognition
Image Segmentation

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report provides a thorough review and mathematical analysis of the alternate Image Algebra parallel research effort conducted by the Singer-Kearfott Corporation. The analysis shows a comparison and the contrasting features of this effort and the Image Algebra developed by the University of Florida. The rationale for the strengths and weakness of the respective algebras are documented and a proposed synthesis of the two efforts is presented.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☑ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Patrick C. Coffield	22b. TELEPHONE (Include Area Code) (904) 882-2838 22c. OFFICE SYMBOL AFATL/AGS

DD Form 1473, JUN 86 *Previous editions are obsolete.*

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

PREFACE

This report describes the effort of synthesizing two distinct image algebra developments, sponsored by the Air Force Armament Laboratory (AFATL) at Eglin Air Force Base (AFB). These algebras were developed independently, one by the Singer-Kearfott Corporation, and the other by the University of Florida. The results of this effort are described in this report. The report was prepared by the Department of Computer and Information Sciences, 301 Computer Science and Engineering Building, University of Florida, Gainesville, Florida 32611, under Air Force Contract F08635-84-C-0295.

The authors wish to thank the Air Force Armament Laboratory and the Defense Advanced Research Projects Agency (DARPA) for sponsoring the development of the image algebra. We are particularly grateful to Dr. Sam Lambert (AFATL), Dr. Donald Daniel (AFATL), Dr. Jasper Lupo (DARPA), Mr. Patrick Coffield (AFATL), and to Mr. Neal Urquhart for their continued support of this research.

TABLE OF CONTENTS

LIST OF FIGURES

STANDARD SYMBOLS OF THE AFATL IMAGE ALGEBRA

Symbol	Explanation
$\mathbf{Z}$	the set of integers
$\mathbf{R}$	the set of real numbers
$\mathbf{C}$	the set of complex numbers
$\mathbf{Z}_{2^k}$	the set of integers with binary representation of length k
$\emptyset$	the empty set
$\in, \notin, \subset$	is an element of, is not an element of, is a subset of
$\cup, \cap$	set union, set intersection
$f:\mathbf{X}\rightarrow\mathbf{Y}$	f is a function from $\mathbf{X}$ to $\mathbf{Y}$
f^{-1}	the inverse of function f
$f\|_{\mathbf{X}}$	the restriction of f to domain $\mathbf{X}$
$\tilde{f}$	the extension of f to a larger domain
$\mathbf{X}_1\times\cdots\times\mathbf{X}_n$	the n-fold cartesian product of sets $\mathbf{X}_1, \cdots, \mathbf{X}_n$
$\prod_{i=1}^{n}\mathbf{X}_i$	the n-fold cartesian product of sets $\mathbf{X}_1, \cdots, \mathbf{X}_n$
$p_j(\mathbf{x}_1, \cdots, \mathbf{x}_n)$	projection of the jth coordinate, $\mathbf{x}_j$, of $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$
$\mathbf{R}_\infty$	the extended real number system
$\vee, \wedge$	maximum, minimum
$\{\mathbf{F}, \mathcal{O}\}$	the algebra with value set $\mathbf{F}$ and operation set $\mathcal{O}$
$\mathbf{F}$	a value set
$\mathcal{O}$	a set of finitary operations

Symbol	Explanation
$o \in O$	o is an operation in set O
$\mathbf{X} \backslash \mathbf{Y}$	the set difference of $\mathbf{X}$ and $\mathbf{Y}$
$\mathbf{W}, \mathbf{X}, \mathbf{Y}$	coordinate sets
$\mathbf{w}, \mathbf{x}, \mathbf{y}$	coordinate points
$\mathbf{F}^{\mathbf{X}}$	the set of all $\mathbf{F}$ valued images on $\mathbf{X}$
$\mathbf{a}, \mathbf{b}, \mathbf{c}$	images
$\mathbf{1}$	a constant image with values at each coordinate 1
$\mathbf{0}$	a constant image with values at each coordinate 0
$\chi_S(\mathbf{a})$	the characteristic function over set S of image $\mathbf{a}$
$f(\mathbf{a})$	the function f induced pointwise over image $\mathbf{a}$
Domain($\mathbf{a}$)	the coordinate set of image $\mathbf{a}$
Range($\mathbf{a}$)	the set of values assigned to coordinates in images $\mathbf{a}$
$\mathbf{r}, \mathbf{s}, \mathbf{t}$	templates $\mathbf{r}$, $\mathbf{s}$, and $\mathbf{t}$, respectively
$\mathbf{t}_y \equiv \mathbf{t}(\mathbf{y})$	the image function of template $\mathbf{t}$
$S(\mathbf{t}_y)$	the support or configuration of $\mathbf{t}_y$
$S_\infty(\mathbf{t}_y)$	the support of $\mathbf{t}_y$ with respect to $\boxtimes$ and $\bigvee$
$(\mathbf{F}^{\mathbf{X}})^{\mathbf{Y}}$	the set of all $\mathbf{F}$ valued templates from $\mathbf{Y}$ to $\mathbf{X}$
$\oplus$	generalized convolution
$\bigvee, \bigwedge$	multiplicative maximum, multiplicative minimum
$\boxtimes$, $\boxtimes$	additive maximum, additive minimum
$\bar{\mathbf{a}}$	the complement of $\mathbf{a}$

Symbol	Explanation
$\mathbf{a}\|_{\mathbf{Y}}$	the restriction of **a** to the domain **Y**
$\mathbf{a}\|\|_{T}$	the restriction of **a** to the set of pixels satisfying property T
$\mathbf{a}\|^{(\mathbf{b},\mathbf{Y})}$	the extension of **a** to **b** on **Y**
choice(S)	the choice function, returning an arbitrary element of set S
card(S)	the cardinality function, counting the number of elements in set S
$\sum\mathbf{a}$	the sum of all pixel values of the image **a**
$\bigvee\mathbf{a}$	the maximum pixel value in image **a**
$\mathbf{a}^{t}$	the transpose of image **a**
q	the injection function $q:\mathbf{F}_k \rightarrow \prod_{i=1}^{n}\mathbf{F}_i$
q_i	the ith-coordinate injection function $q_i:\prod_{j=1}^{n}\mathbf{F}_j \rightarrow \prod_{j=1}^{n}\mathbf{F}_j$, defined by $q_i \equiv qp_i$

SECTION I
INTRODUCTION

The Air Force Armament Laboratory (AFATL) is developing autonomous seekers to attack both mobile land targets and fixed, high-value land targets. The purpose of the mathematical structure described in this report is to aid the efficient development of such seekers. This mathematical structure, known as the AFATL Image Algebra, has been specifically designed for the concise expression and clear representation of image processing and pattern recognition techniques. This structure provides a common mathematical environment for target detection, algorithm development, optimization, comparison, coding, and performance evaluation. In addition, the Image Algebra provides a mathematical basis for a universal image processing language which, when properly implemented, will greatly increase a researcher's productivity as programming tasks required to compute image transformations are greatly simplified due to the replacement of large blocks of code by concise algebraic expressions.

Several previous attempts to develop a unified algebraic approach to image processing have met only partial success in expressing all transformations of gray value images, Reference 1. In contrast, in addition to meeting the design specifications mentioned in the previous paragraph, the AFATL Image Algebra provides a complete unified algebraic structure capable of expressing all image-to-image transformations. The Image Algebra's foundation evolved from a 33-month Air Force/Defense Advanced Research Project Agency (DARPA) sponsored research effort known as the Image Algebra Project. Two contracts to develop the Image Algebra were awarded. One award went to the University of Florida, with Dr. Gerhard Ritter as principal investigator, and the other to the Singer-Kearfott Corporation with Dr. Charles Giardina as principal investigator.

The University of Florida's Image Algebra development proved highly successful, capable of fulfilling the tasks set forth by the Air Force Armament Division. In order to further enhance the value of the algebra, the Air Force Armament Laboratory awarded a contract extension to the University of Florida with the goal of providing an Image Algebra Fortran implementation and to investigate a possible synthesis of the structures developed by the Singer-Kearfott Corporation and University of Florida.

This report summarizes the University of Florida's analysis of the two algebraic structures and provides a tutorial overview of the AFATL Image Algebra.

SECTION II

ANALYSIS OF SINGER-KEARFOTT'S IMAGE ALGEBRA

It became clear at the December 1985 DARPA/Center for Night Vision and Electro-Optics (CNVEO) Image Algebra briefing by the University of Florida and Singer-Kearfott that two vastly divergent algebras were being formulated. The University of Florida's algebra was viewed as the more mature and superior of the two, and the suggestion was made to continue the effort with the University of Florida as sole contractor. However, AFATL program management decided to retain both contractors.

One of the major differences of the two approaches was that Singer-Kearfott's algebra had an infinite number of operators while the University of Florida's had 10. After this meeting, Singer-Kearfott's operands and operators started, to some extent, mimicking the operands and operators established at the University of Florida. However, as the University of Florida's operands and operators became more refined, many of Singer-Kearfott's operands and operators retained much of the appearance, characteristics, and properties of several of the University of Florida's earlier defined operands and operators. In particular, Singer-Kearfott's images are elements of $\mathbf{X} = \bigcup_{A \in Z^2} \mathbf{R}^A$, where $\mathbf{Z}^2$ denotes the infinite planar array with integral coordinates, $\mathbf{R}$ the set of real numbers, and $\mathbf{R}^A$ denotes the set of all functions from A to $\mathbf{R}$. As we will show later on, this was the University of Florida's early definition of real valued images. On the other hand, images in the University of Florida's algebra are elements of $\mathbf{F}^X$, where $\mathbf{X} \subset \mathbf{R}^k$, $\mathbf{R}^k$ denotes euclidean k-dimensional space, and $\mathbf{F}$ a set of values such as the set of real numbers, complex numbers, binary numbers of fixed length k, etc. This allows for complex images as well as discretized computer images. As Singer-Kearfott's images do not include complex imagery, we do not see how any Fourier-like image transform can be accomplished within their setting.

As two Singer-Kearfott images need not have the same underlying array A, a simple operation such as the addition of two images becomes a complicated issue, both in terms of definition as well as possible software implementation. Specifically, the fundamental operators in the Singer-Kearfott algebra are:

I	Addition	$\oplus$
II	Multiplication	$\odot$
III	Maximum	$\bigvee$
IV	Division	$\oplus$
V	Translation	T
VI	Rotation	N
VII	Reflection	D
VIII	Domain Extractor	K
IX	Parameter Extractor	G
X	Existential Operator	E

Singer-Kearfott's image addition $\oplus$ is defined as:

$$(f \oplus g) = \begin{cases} f(x,y) & (x,y) \in A - B \\ g(x,y) & (x,y) \in B - A \\ f(x,y) + g(x,y) & (x,y) \in A \cap B \\ \text{undefined} & (x,y) \notin A \cup B \end{cases}$$

Image multiplication $\odot$, maximum $\bigvee$, and division $\oplus$ are defined in an analogous fashion with $+$ being replaced by $\cdot$, max, and $+$, respectively, in the above definition.

1. REMARKS AND OBSERVATIONS

We now consider each of the basic operations of the Singer-Kearfott algebra and discuss them in the context of image processing applications and the corresponding operations of the University of Florida algebra.

a. Image Addition, Maximum, and Division

Image addition, maximum, and division constitute the basic binary image operations in the Singer-Kearfott algebra. The University of Florida had defined this type of image addition, multiplication, maximum, and division at the beginning of Phase I of the project. The next page shows a viewgraph dating from the 2nd project review of Phase I (March 27, 1985).

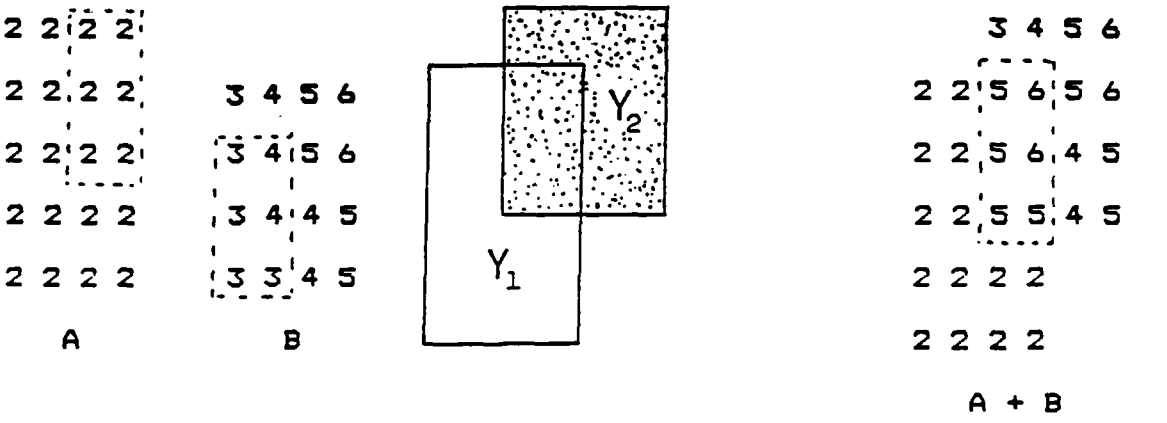
Viewgraph of the 2nd project review of Phase I (March 27, 1985):

IMAGE ALGEBRA PROJECT

THE OPERATIONS +, *, ∨, ∧ ON S_X

LET A BE AN IMAGE ON Y_1 AND B BE AN IMAGE ON Y_2, WHERE $Y_1, Y_2 \in P(X)$

1. $A + B = \{ (x,c(x)) : c(x) = a(x) + b(x),\ x \in Y_1 \cap Y_2 \} \cup A_B \cup B_A$



SIMILARLY

2. $A * B = \{ (x,c(x)) : c(x) = a(x) * b(x),\ x \in Y_1 \cap Y_2 \} \cup A_B \cup B_A$

3. $A \vee B = \{ (x,c(x)) : c(x) = a(x) \vee b(x),\ x \in Y_1 \cap Y_2 \} \cup A_B \cup B_A$

4. $A \wedge B = \{ (x,c(x)) : c(x) = a(x) \wedge b(x),\ x \in Y_1 \cap Y_2 \} \cup A_B \cup B_A$

b. Critique of Basic Binary Operands

The University of Florida abandoned these definitions fairly early as they are too clumsy from both the application and theoretical point of view and the image processing community simply does not add images this way. In addition, there exist no natural additive and multiplicative identities for these operations (i.e. a **0** and a **1** image). The only identity that works for both is the empty set. This does not provide for a very useful mathematical structure.

c. Corresponding University of Florida Operations

The Singer-Kearfott addition (multiplication, maximum, division), if ever needed, can be easily expressed in terms of the University of Florida's current addition (multiplication, maximum, division) followed by the extension operator.

d. Image Rotation

Singer-Kearfott's rotation is defined only for multiples of 90 degrees about the origin. In the University of Florida's algebra this can be accomplished through image definition (i.e. the assignment statement **a**(j,-i) = **a**(i,j)) or with a template operation. Furthermore, rotation using a template is a one-step operation and images can be rotated through any angle and about any point.

e. Image Translation and Reflection

The operations of translation and reflection as defined in the Singer-Kearfott algebra can again be accomplished by a simple assignment statement or a one step template operation. Furthermore, Singer-Kearfott's translation is a simple linear coordinate shift while translation by templates encompasses both linear and non-linear translations.

f. Domain Extractor and Parameter Extractor

The domain extractor and parameter extractor correspond to the University of Florida's operation of domain and range, respectively.

g. Existential Operator

In regard to the existential operator, we prefer to adhere to standard textbook mathematics and leave it as a function definition statement (e.g. "Let **a** be the function defined by...") instead of viewing it as an algebraic operation. Our view is identical with computer language implementations in general. Functions that do not exist in a particular language are defined by assignment statements (e.g. **a**(i,j) = i+j, etc.).

h. Remaining Operators

The remaining operators consists of an infinite number of macro operators

(subroutines) which are defined in terms of the 10 elementary operations. This approach makes even the simplest algorithm appear extremely unreadable and cumbersome to express. Two examples should suffice.

(1) Morphological Operations

The simple operations of dilation, erosion, opening and closing have the following ungainly appearance:

$$[\hat{D}\hat{I}(f,g)](i,j) = E[V_o\{\mathcal{A}(f, g_{i,j})\}, (i,j)]$$

$$[\hat{E}\hat{R}(f,g)](i,j) = E[V_o\{\mathcal{A}(f, \ominus g_{i,j})\}, (i,j)]$$

$$\hat{O}\hat{P}(f,g) = \hat{E}\hat{R}[\hat{D}\hat{I}\{f, N^2(g)\}, N^2(g)]$$

$$\hat{C}\hat{L}(f,g) = \hat{D}\hat{I}[\hat{E}\hat{R}(f,g), g]$$

where

$$V_o(f) = G[\bigvee_{(i,j)\in A} \mathcal{S}[T_{-i,-j}(f), \{(0,0)\}]]$$

$$\mathcal{A}(f,g) = \oplus[\oplus([f \oplus g] \odot [1_{A\cap B} \oplus 0_{A\cup B}])] \oplus 0_{A\cap B}$$

$$\mathcal{S}(f,B) = M(f,1_B), \text{ and } M(f,g) = \mathcal{A}(f \odot g, 0_{A\cap B})$$

In addition, g is assumed to be symmetric.

(2) The Gradient Operator

The explanation of the common gradient operator is as follows:

$$E_\infty(f,M,N,t) = \mathcal{T}_t[N_\infty\{\mathcal{F}(f,M), \mathcal{F}(f,N)\}]$$

$$E_1(f,M,N,t) = \mathcal{T}_t[N_1\{\mathcal{F}(f,M), \mathcal{F}(f,N)\}]$$

$$E_2(f,M,N,t) = \mathcal{T}_t[N_2\{\mathcal{F}(f,M), \mathcal{F}(f,N)\}]$$

In each of the above, M and N are images which, in practice represent directional masks. Here

$$N_\infty(f_1,f_2,\ldots,f_m) = \bigvee_{k=1}^{m} |f_k|$$

$$N_1(f_1,f_2,\ldots,f_m) = \sum_{k=1}^{m} |f_k|$$

$$N_2(f_1,f_2,\ldots,f_m) = \{\sum_{k=1}^{m} [f_k]^2\}^{\frac{1}{2}}$$

$$\mathcal{F}(f,g) = \sum_{(i,j)\in A} [E[\mathcal{D}_o[\mathcal{R}(f,T_{i,j}(g)), T_{i,j}(g)], \{(i,j)\}]$$

where $\sum_{(i,j)\in A}$ denotes $\oplus$ addition,

$$\mathcal{D}_o(f,g) = \sum_o[f \odot g], \qquad \sum_o(f) = G[\sum_{(i,j)\in A} S[T_{-i,-j}(f), \{(0,0)\}]]$$

$$\mathcal{T}_o(f) = [(f \oslash 0_A) \oplus_o (f \oslash 0_A)]^c, \qquad \oplus_o f = (\oplus f) \oplus 0_A$$

$$\mathcal{T}_t = \mathcal{T}_o(f \ominus t \Delta 1_A), \qquad \text{where } 0_A = E[0,K(f)]$$

and

$$\alpha \Delta f = E(\alpha,K(f)) \odot f.$$

2. SUMMARY

Overall, we found the Singer-Kearfott algebra extremely cumbersome and ungainly, not in step with much of current image processing practice. The algebra is extremely difficult to use in expressing even the simplest algorithm, let alone an algorithm of any major complexity. This defeats at least one major goal of AFATL's image processing language development project, namely simplicity and translucency of algorithm representation.

In all fairness to the Singer-Kearfott effort we must add that our conclusions are based on very scant reference material. Our reference material consisted of Singer's Phase I final report and the Phase II final review viewgraphs. Singer-Kearfott's final report, which was requested by us and the intended basis of this analysis and synthesis, was never forwarded by AFATL. Both, Singer's Phase I final report and the Phase II final review viewgraphs are sorely lacking algorithmic image algebra expressions , i.e. image algebra pseudo code of algorithms that represent more than just a simple transform or maximum operation.

In summary, the University of Florida's image algebra is capable of expressing Singer-Kearfott's elemental operators in a straight forward fashion and thus encompasses all capabilities of the Singer-Kearfott algebra. In addition, considering the shortcommings of the

Singer-Kearfott algebra, we feel that any synthesis of the two algebraic structures would be counter-productive to the goals of the AFATL image processing language development project. We therefore recommend the algebraic structure developed by the University of Florida to represent the AFATL standard image algebra.

SECTION III
IMAGE ALGEBRA

In the remaining part of this document we provide an overview of the AFATL Standard Image Algebra. We begin by introducing the basic operands and operators of the algebra.

1. VALUE SETS AND COORDINATE SETS

The image algebra defined in this paper is capable of multivalue image manipulation and constitutes a heterogeneous algebra in the sense of Birkhoff, Reference 2. This general form of the image algebra is discussed in the last sections of this document. In this section we restrict ourselves to single-valued image manipulation.

The image algebra deals with six basic types of operands, namely value sets, coordinate sets, the elements of each of these sets, images, and templates. The only value sets considered in this section are the set of integers, real numbers, extended real numbers (which include the symbols $+\infty$ and $-\infty$), complex numbers, and binary numbers of fixed length k, which will be denoted by $\mathbf{Z}$, $\mathbf{R}$, $\mathbf{R}_{\infty}$, $\mathbf{C}$, and $\mathbf{Z}_{2^k}$, respectively. These correspond to the values commonly encountered in most image processing routines and in the modeling of such routines.

An unspecified value set will henceforth be denoted by $\mathbf{F}$. The operations on and between elements of a given value set $\mathbf{F} \in \{\mathbf{Z}, \mathbf{R}, \mathbf{R}_{\infty}, \mathbf{C}, \mathbf{Z}_{2^k}\}$ are the usual elementary operations associated with $\mathbf{F}$. Thus, if $\mathbf{F} = \mathbf{R}$, then the operations are the usual arithmetic and logic operations of addition, multiplication and maximum, and the complementary operations of subtraction, division, and minimum. In addition to these elementary operations on elements of value sets, the image algebra also includes the operations of union, intersection, set subtraction, choice function, and cardinality function on subsets of $\mathbf{F}$. The choice function applied to a set returns (chooses) an arbitrary element of the set, while the cardinality function yields the number of elements in the set. Union, intersection, set subtraction, choice function, and cardinality function will be denoted by $\cup$, $\cap$, $\setminus$, choice, and card, respectively.

Coordinate sets are subsets of n-dimensional Euclidean space $\mathbf{R}^n$. We reserve the letters $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{W}$ to denote coordinate sets. Elements of coordinate sets will be denoted by bold lower case letters. In particular, if $\mathbf{x} \in \mathbf{X}$ and $\mathbf{X} \subset \mathbf{R}^n$, then $\mathbf{x}$ is of form

$\mathbf{x}=(x_1, x_2, \cdots, x_n)$, where each coordinate x_i (i = 1,2, ... ,n) is a real number.

It follows from the definition that coordinate sets can be rectangular, hexagonal, toroidal discrete arrays as well as infinite subsets of $\mathbf{R}^n$. Providing coordinate sets with such wide varieties of shapes, sizes and dimensions allows for a coherent mathematical approach to the modeling and manipulation of continuous as well as discrete images on any desired type of coordinate set.

Image algebra operations acting on coordinate sets are operations on subsets of coordinate sets as well as operations between coordinate points. In particular, operations on subsets of coordinate sets are $\cup$, $\cap$, $\setminus$, choice function, and cardinality function. Image algebra operations on or between elements of coordinate sets are the usual operations between coordinate points, i.e., vector addition, scalar and vector multiplication, dot product, etc.

2. IMAGES

Thus far we have defined two types of objects, value sets and coordinate sets. These sets and their elements constitute some important operands of the image algebra. However, the most fundamental of the algebra's operands are images. The most general, yet useful, mathematical definition of an image involves the previously defined concepts of value sets and coordinate sets.

Given a coordinate and value sets $\mathbf{X}$ and $\mathbf{F}$, respectively, then an $\mathbf{F}$ Ivalued image $\mathbf{a}$ on $\mathbf{X}$ is the graph of a function $\mathbf{a}:\mathbf{X} \rightarrow \mathbf{F}$. Thus, an $\mathbf{F}$ valued image $\mathbf{a}$ on $\mathbf{X}$ is of the form:

$$\mathbf{a} = \{ (\mathbf{x},\mathbf{a}(\mathbf{x})) : \mathbf{x} \in \mathbf{X} \},$$

where $\mathbf{a}(\mathbf{x}) \in \mathbf{F}$.

The set $\mathbf{X}$ is called the set of image coordinates of $\mathbf{a}$, and the range of the function $\mathbf{a}$ (which is a subset of $\mathbf{F}$) is the set of image values of $\mathbf{a}$. An element $(\mathbf{x},\mathbf{a}(\mathbf{x}))$ of the image $\mathbf{a}$ is called a picture element or pixel, where $\mathbf{x}$ is called the pixel location, and $\mathbf{a}(\mathbf{x})$ the pixel value at location $\mathbf{x}$. The set of all $\mathbf{F}$ valued images on $\mathbf{X}$ is denoted by $\mathbf{F}^{\mathbf{X}}$. Here we follow the usual mathematical convention of denoting the set of all functions from a set A to a set B by B^A.

If the value set $\mathbf{F} = \mathbf{Z}$ or $\mathbf{F} = \mathbf{R}$, then we are dealing with integer or real valued images, respectively. Similarly, replacing $\mathbf{F}$ by $\mathbf{C}$, or $\mathbf{Z}_{2^k}$, provides for complex or finite digital images, respectively.

In the next chapter we extend the notion of value sets in order to include such objects as vector valued images. This fact and the above examples should make it clear that the various choices for $\mathbf{F}$ and $\mathbf{X}$ allow for a far greater variety of image operands than are currently used by the image processing community.

3. BINARY AND UNARY OPERATIONS ON IMAGES

Operations on and between $\mathbf{F}$ valued images are the natural induced operations of the algebraic system $\mathbf{F}$. For example, for real valued images (i.e. elements of $\mathbf{R}^{\mathbf{X}}$), the operations are the elementary operations induced by the vector lattice (a vector space which is also a lattice) $\mathbf{R}$. Thus, the basic real valued image operations reflect the arithmetic and logic operations on $\mathbf{R}$. In particular, the binary operations of addition, multiplication, and maximum on $\mathbf{R}^{\mathbf{X}}$ are defined as follows:

Let $\mathbf{a}, \mathbf{b} \in \mathbf{R}^{\mathbf{X}}$. Then

$$\mathbf{a} + \mathbf{b} \equiv \{ (\mathbf{x},\mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) + \mathbf{b}(\mathbf{x}), \mathbf{x} \in \mathbf{X}\} \tag{1}$$

$$\mathbf{a} * \mathbf{b} \equiv \{ (\mathbf{x},\mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \cdot \mathbf{b}(\mathbf{x}), \mathbf{x} \in \mathbf{X}\} \tag{2}$$

$$\mathbf{a} \vee \mathbf{b} \equiv \{ (\mathbf{x},\mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \vee \mathbf{b}(\mathbf{x}), \mathbf{x} \in \mathbf{X}\} \tag{3}$$

These are the basic binary operations for real valued images. As complex numbers are not endowed with a "natural" lattice structure, only operations 1 and 2 are basic operations between complex valued images.

Analogous to the development of the algebra of real numbers, other binary and unary operations on real valued images can now be derived from the basic operations either directly or in terms of series expansion. However, instead of reinventing the wheel, we assume familiarity with the algebra of real numbers and let the remaining operations on $\mathbf{R}^{\mathbf{X}}$ again be induced by the corresponding operations on $\mathbf{R}$. Two of these operations - commonly used in image processing - are exponentiation and the computation of logarithms. In particular, if $\mathbf{a}$ and $\mathbf{b}$ are real valued images on $\mathbf{X}$, then

$$\mathbf{a}^{\mathbf{b}} \equiv \{(\mathbf{x},\mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x})^{\mathbf{b}(\mathbf{x})} \text{ if } \mathbf{a}(\mathbf{x}) \neq 0, \text{ otherwise } \mathbf{c}(\mathbf{x}) = 0, \mathbf{x} \in \mathbf{X}\}. \tag{4}$$

As we are dealing with real valued images we follow the rules of real arithmetic and restrict this binary operation to those pairs of images $\mathbf{a},\mathbf{b}$ for which $\mathbf{a}(\mathbf{x})^{\mathbf{b}(\mathbf{x})} \in \mathbf{R}$ whenever

$\mathbf{a}(\mathbf{x}) \neq 0$. This prevents the creation of complex pixel values such as $(-1)^{1/2}$. The inverse of exponentiation is defined in the usual way by taking the logarithm, namely

$$\log_{\mathbf{a}}\mathbf{b} \equiv \{(\mathbf{x},\mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \log_{\mathbf{a}(\mathbf{x})}\mathbf{b}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}. \tag{5}$$

As for real numbers, $\log_{\mathbf{a}}\mathbf{b}$ is defined only for those images $\mathbf{a}$ and $\mathbf{b}$ for which $\mathbf{a}(\mathbf{x}) > 0$ and $\mathbf{b}(\mathbf{x}) > 0$ for all $\mathbf{x} \in \mathbf{X}$. The next basic binary operation, called the dot product, distinguishes itself from the above five in that its output is not an image but a real number. Let $\mathbf{X}$ be finite, then the dot product is defined as:

$$\mathbf{a} \bullet \mathbf{b} \equiv \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{a}(\mathbf{x}) \cdot \mathbf{b}(\mathbf{x}) \tag{6}$$

An image $\mathbf{a}$ is called a constant image if all its gray values are the same; i.e. if $\mathbf{a}(\mathbf{x}) = k$ for some real number k and for all $\mathbf{x} \in \mathbf{X}$.

Two constant images are of utmost importance in the image algebra; these are the zero image, defined by $\mathbf{0} \equiv \{(\mathbf{x},0) : \mathbf{x} \in \mathbf{X}\}$, and the unit image, defined by $\mathbf{1} \equiv \{(\mathbf{x},1) : \mathbf{x} \in \mathbf{X}\}$.

Suppose $k \in \mathbf{R}$ and $\mathbf{a}$ is a constant image with $\mathbf{a}(\mathbf{x}) = k$. Then we define:

$\mathbf{b}^{\mathbf{k}} \equiv \mathbf{b}^{\mathbf{a}}$ and $k^{\mathbf{b}} \equiv \mathbf{a}^{\mathbf{b}}$

$k\mathbf{b} \equiv \mathbf{a} * \mathbf{b}$ and $k + \mathbf{b} \equiv \mathbf{a} + \mathbf{b}$

$\log_{\mathbf{k}}\mathbf{b} \equiv \log_{\mathbf{a}}\mathbf{b}$.

In the definition of log we assume, of course, that $k > 0$ and $\mathbf{b}(\mathbf{x}) > 0$ for all $\mathbf{x}$. We also note that exponentiation is defined even when $\mathbf{a}(\mathbf{x}) = 0$.

Subtraction, division and minimum are defined in terms of the basic operations and inverses. Specifically:

$\mathbf{a} - \mathbf{b} \equiv \mathbf{a} + (-\mathbf{b})$ and $\mathbf{a}/\mathbf{b} \equiv \mathbf{a} * \mathbf{b}^{-1}$, where $-\mathbf{b} = \{(\mathbf{x}, -\mathbf{b}(\mathbf{x})) : (\mathbf{x},\mathbf{b}(\mathbf{x})) \in \mathbf{b}\}$

$\mathbf{a} \wedge \mathbf{b} \equiv -(-\mathbf{a} \vee -\mathbf{b})$

The images $\mathbf{0}$ and $\mathbf{1}$ have the obvious property $\mathbf{a} + \mathbf{0} = \mathbf{a}$ and $\mathbf{a} * \mathbf{1} = \mathbf{a}$. On the other hand, $\mathbf{b} * \mathbf{b}^{-1}$ does not necessarily equal $\mathbf{1}$. However, $\mathbf{b} * \mathbf{b}^{-1} * \mathbf{b} = \mathbf{b}$. For this reason $\mathbf{b}^{-1}$ is called the pseudo inverse of $\mathbf{b}$. Inequalities between images are defined in terms of maximum and minimum. Thus, for example, $\mathbf{a} \leq \mathbf{b}$ if and only if $\mathbf{a} \vee \mathbf{b} = \mathbf{b}$. These observations show that the ring $(\mathbf{R}^{\mathbf{X}}, +, *)$ and the lattice $(\mathbf{R}^{\mathbf{X}}, \vee, \wedge)$ behave very much like the ring and lattice of real numbers.

There are various useful unary operations definable in terms of the basic binary operations. For example, we have already provided the definitions of $\mathbf{k}^{\mathbf{b}}$ and $\log_{\mathbf{k}}\mathbf{b}$, the exponential of an image and the logarithm of an image $\mathbf{b}$ to the base k, respectively. In particular, the exponential of an image $\mathbf{b}$, $\exp(\mathbf{b}) = \mathrm{e}^{\mathbf{b}} \equiv \mathbf{a}^{\mathbf{b}}$, and the natural logarithm of $\mathbf{b}$ is defined as $\ln\mathbf{b} \equiv \log_{\mathbf{a}}\mathbf{b}$, respectively, where $\mathbf{a}$ is the constant image defined by $\mathbf{a}(\mathbf{x}) \equiv \mathrm{e}$ for all $\mathbf{x} \in \mathbf{X}$. Similarly, the absolute value of an image $\mathbf{a}$ can be defined by $|\mathbf{a}| \equiv \mathbf{a} \vee (-\mathbf{a})$.

In view of these examples it is obvious that there are various operations on and between images of different degrees of complexity that can be derived from the operations defined thus far. Phrasing some of these in terms of the basic operations would result in complicated algebraic expressions or infinite series representation. However, this would defeat the goal of providing a simple language for image processing tasks. Instead, we follow our initial philosophy and define the operations on $\mathbf{F}^{\mathbf{X}}$ to be the operations induced by the (usually well-known) operations of the algebraic system $\mathbf{F}$. In particular, the common unary operations on $\mathbf{R}^{\mathbf{X}}$ are functions available in most high level programming languages. More generally, any function $f\colon \mathbf{R} \to \mathbf{R}$ induces a function $\mathbf{R}^{\mathbf{X}} \to \mathbf{R}^{\mathbf{X}}$, again denoted by f, and defined by

$$f(\mathbf{a}) = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = f(\mathbf{a}(\mathbf{x}))\}$$

For example, $\sin(\mathbf{a}) = \{(\mathbf{x}, \sin(\mathbf{a}(\mathbf{x}))) : \mathbf{x} \in \mathbf{X}\}$. Similarly, if χ_S denotes the characteristic function with respect to some set $S \subset \mathbf{R}$, then

$$\chi_S(\mathbf{a}) = \{(\mathbf{x}, \chi_S(\mathbf{a}(\mathbf{x}))) : \mathbf{x} \in \mathbf{X}\} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) \in S, \text{ otherwise } \mathbf{c}(\mathbf{x}) = 0\}.$$

Excepting the formalism, many of the operations of the algebraic system $\mathbf{R}^{\mathbf{X}}$ described thus far are not new, and are well-known to the image processing community. What is new is the concept of raising an image to the power of another image or taking the logarithm of an image to the base of another image. This generalizes the common pixel level operations of raising an image to a constant such as taking the square root of an image at each pixel or taking the natural logarithm of an image at each pixel.

The idea of having more versatile pixel level operations has led to the following generalization of the characteristic function: Given $\mathbf{a} \in \mathbf{F}^{\mathbf{X}}$ and $S \in (2^{\mathbf{F}})^{\mathbf{X}}$, where $2^{\mathbf{F}}$ denotes the power set of $\mathbf{F}$ so that $S(\mathbf{x}) \subset \mathbf{F}$ for each $\mathbf{x} \in \mathbf{X}$, then

$$\chi_S(\mathbf{a}) = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) \in S(\mathbf{x}), \text{ otherwise } \mathbf{c}(\mathbf{x}) = 0\}. \tag{7}$$

Pixel level image comparison provides a simple application example of the generalized

characteristic function. Given the image $\mathbf{b} \in \mathbf{R}^{\mathbf{X}}$, we define $S_{\leq \mathbf{b}} \in (2^{\mathbf{R}})^{\mathbf{X}}$ by $S_{\leq \mathbf{b}}(\mathbf{x}) \equiv \{r \in \mathbf{R} : r \leq \mathbf{b}(\mathbf{x})\}$. The functions $S_{<\mathbf{b}}$, $S_{=\mathbf{b}}$, $S_{\geq \mathbf{b}}$, and $S_{>\mathbf{b}}$ are defined analogously. Thus, for example, $S_{>\mathbf{b}}(\mathbf{x}) \equiv \{r \in \mathbf{R} : r > \mathbf{b}(\mathbf{x})\}$ and $S_{=\mathbf{b}}(\mathbf{x}) \equiv \{r \in \mathbf{R} : r = \mathbf{b}(\mathbf{x})\}$. Substituting these set functions for S in Equation (7) yields:

$$\chi_{S_{>\mathbf{b}}}(\mathbf{a}) = \{ (\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) > \mathbf{b}(\mathbf{x}), \text{ else } \mathbf{c}(\mathbf{x}) = 0 \},$$

$$\chi_{S_{\leq \mathbf{b}}}(\mathbf{a}) = \{ (\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) \leq \mathbf{b}(\mathbf{x}), \text{ else } \mathbf{c}(\mathbf{x}) = 0 \}, \text{ etc.}$$

In order to reduce and simplify notation, we define $\chi_{>\mathbf{b}} \equiv \chi_{S_{>\mathbf{b}}}$, $\chi_{\leq \mathbf{b}} \equiv \chi_{S_{\leq \mathbf{b}}}$, $\chi_{<\mathbf{b}} \equiv \chi_{S_{<\mathbf{b}}}$, $\chi_{\geq \mathbf{b}} \equiv \chi_{S_{\geq \mathbf{b}}}$, and $\chi_{\mathbf{b}} \equiv \chi_{S_{=\mathbf{b}}}$. As alluded to earlier, these generalized characteristic functions could have been defined in terms of the more elementary image operations. Specifically, we could have defined

$$\chi_{>\mathbf{b}}(\mathbf{a}) = [(\mathbf{a} - \mathbf{b}) \vee \mathbf{0}]^{-1} * [(\mathbf{a} - \mathbf{b}) \vee \mathbf{0}]$$

Obviously, if $\mathbf{a} = \mathbf{b}$, then $(\mathbf{a} - \mathbf{b}) \vee \mathbf{0} = \mathbf{0}$ and $\chi_{>\mathbf{b}}(\mathbf{a}) = \mathbf{0}^{-1} * \mathbf{0} = \mathbf{0}$ since by definition of exponentiation, $\mathbf{0}^{-1} = \mathbf{0}$. The function $\chi_{<\mathbf{b}}$ is defined in a similar fashion. The remaining characteristic functions that compare two images can then be defined in terms of products and complementations. In particular, if we define the complement $\bar{\mathbf{a}}$ of an image $\mathbf{a}$ by $\bar{\mathbf{a}} \equiv \mathbf{1} - \mathbf{a} * \mathbf{a}^{-1}$, then we obtain

$$\chi_{\leq \mathbf{b}}(\mathbf{a}) = \overline{\chi_{>\mathbf{b}}(\mathbf{a})}, \quad \chi_{\geq \mathbf{b}}(\mathbf{a}) = \overline{\chi_{<\mathbf{b}}(\mathbf{a})}, \quad \text{and} \quad \chi_{\mathbf{b}}(\mathbf{a}) = \chi_{\leq \mathbf{b}}(\mathbf{a}) * \chi_{\geq \mathbf{b}}(\mathbf{a}).$$

Whenever $\mathbf{b}$ is the constant image with gray values equal to k it is customary to replace $\mathbf{b}$ by k in the above definitions. Figure 1 below provides an example of the operation $\chi_{\geq \mathbf{k}}(\mathbf{a})$, where $k = 15$. In a likewise fashion we can retain pixels whose values are within an interval $[m,n]$ by using

$$\chi_{[m,n]}(\mathbf{a}) = \chi_{\geq m}(\mathbf{a}) * \chi_{\leq n}(\mathbf{a}).$$

Figure 1. Example of the Operation $\chi_{\geq 15}(\mathbf{a})$

When considering an expression of the form sin(**a**), one rarely thinks of the image **a** as a function. However, images, as defined in this report, are functions, namely elements of $\mathbf{F}^{\mathbf{X}}$, and several important mathematical notions used in image processing are the restriction, extension, domain, and range of a function. We express these notions as basic operations of the image algebra. In particular, if **a** is an image on **X**, then the Domain function of an image **a** is defined simply as

$$\text{Domain}(\mathbf{a}) = \text{set over which } \mathbf{a} \text{ is defined,}$$

e.g., $\text{Domain}(\mathbf{a}) = \mathbf{X}$. The range function of an image lies on the other side of the spectrum, and it provides the set of values assumed by the image. Thus, if $\mathbf{a} \in \mathbf{F}^{\mathbf{X}}$, then

$$\text{Range}(\mathbf{a}) = \text{set of values determined by } \mathbf{a},$$

e.g., $\text{Range}(\mathbf{a}) \subset \mathbf{F}$ is the set of all values **a** assumes on **X**. Therefore, the output of domain or range is not an image array but a set of coordinate points or a set of values, respectively.

Let $\mathbf{a} \in \mathbf{F}^{\mathbf{X}}$. The restriction of **a** to a subset **Y** of **X** is denoted by $\mathbf{a}|_{\mathbf{Y}}$. Thus, $\mathbf{a}|_{\mathbf{Y}} \in \mathbf{F}^{\mathbf{Y}}$. Here a user would specify the coordinate set $\mathbf{Y} \subset \mathbf{X}$. As an example, he could set $\mathbf{Y} = \{\mathbf{x} \in \mathbf{X} : 5 \leq |\mathbf{x}| \leq 20\}$. We also allow the restriction of **a** to a subset of **X** specified by an image-dependent property such as $\mathbf{Y} = \{\mathbf{x} \in \mathbf{X} : \mathbf{a}(\mathbf{x}) \in \mathrm{S}\}$, where $\mathrm{S} \subset \mathbf{F}$. This type of restriction is denoted by a double vertical bar and provides a useful tool for expressing various algorithmic procedures. For example, if $\mathbf{Y} = \{\mathbf{x} \in \mathbf{X} : \mathbf{a}(\mathbf{x}) \geq \mathrm{T}\}$, where T denotes a given threshold value, then we define $\mathbf{a}\|_{\geq \mathrm{T}}$ by $\mathbf{a}\|_{\geq \mathrm{T}} \equiv \mathbf{a}|_{\{\mathbf{x} \in \mathbf{X} : \mathbf{a}(\mathbf{x}) \geq \mathrm{T}\}}$. In this case, note that

Domain($\mathbf{a}_{|\geq T}$) is the set of all locations where **a** exceeds the threshold T.

Let **a** be an image on **X**, **b** an image on **Y**, and $\mathbf{X} \subset \mathbf{Y}$. The extension of **a** to **b** on **Y** is defined by

$$\mathbf{a}^{|(\mathbf{b},\mathbf{Y})}(\mathbf{x}) = \begin{cases} \mathbf{a}(\mathbf{x}) \text{ if } \mathbf{x} \in \mathbf{X} \\ \mathbf{b}(\mathbf{x}) \text{ if } \mathbf{x} \in \mathbf{Y} \backslash \mathbf{X}, \end{cases}$$

where $\mathbf{Y} \backslash \mathbf{X} = \{\mathbf{y} \in \mathbf{Y} : \mathbf{y} \notin \mathbf{X}\}$. In actual practice, the user will have to specify the function (image) **b** on **Y**.

4. GENERALIZED TEMPLATES

In terms of image processing applications, templates and template operations are the most powerful tool of the image algebra. The image algebra definition of a template unifies and generalizes the usual concepts of templates, masks, windows and neighborhood functions into one general mathematical entity. In addition, templates as defined in this report generalize the notion of "structuring elements" as used in mathematical morphology, Reference 3.

Let **X** and **Y** be coordinate sets and **F** a value set. A generalized **F** valued template **t** from **Y** to **X** is a function $\mathbf{t}: \mathbf{Y} \rightarrow \mathbf{F}^{\mathbf{X}}$. Thus, for each $\mathbf{y} \in \mathbf{Y}$, $\mathbf{t}(\mathbf{y}) \in \mathbf{F}^{\mathbf{X}}$, or, equivalently, $\mathbf{t}(\mathbf{y})$ is an **F** valued image on **X**. The set **Y** is called the target array, and the set **X** is called the source array of the template **t**. For notational convenience we define $\mathbf{t}_\mathbf{y} \equiv \mathbf{t}(\mathbf{y})$. Thus, $\mathbf{t}_\mathbf{y} = \{(\mathbf{x}, \mathbf{t}_\mathbf{y}(\mathbf{x})) : \mathbf{x} \in \mathbf{X}\}$. The point **y** is called the target point of the template **t**, and the values $\mathbf{t}_\mathbf{y}(\mathbf{x})$ are called the weights of the template **t**.

The set of all **F** valued templates from **Y** to **X** will, henceforth, be denoted by $(\mathbf{F}^{\mathbf{X}})^{\mathbf{Y}}$. If **t** is a real valued template from **Y** to **X**, then we define the support of $\mathbf{t}_\mathbf{y}$ to be $\mathcal{S}(\mathbf{t}_\mathbf{y}) = \{ \mathbf{x} \in \mathbf{X} : \mathbf{t}_\mathbf{y}(\mathbf{x}) \neq 0 \}$. If **t** is an extended real valued template then we also define $\mathcal{S}_\infty(\mathbf{t}_\mathbf{y}) = \{ \mathbf{x} \in \mathbf{X} : \mathbf{t}_\mathbf{y}(\mathbf{x}) \neq \pm\infty \}$. The sets $\mathcal{S}(\mathbf{t}_\mathbf{y})$ and $\mathcal{S}_\infty(\mathbf{t}_\mathbf{y})$ are also referred to as the configuration of **t** at target pixel **y** or simply as the source configuration. Figure 2 illustrates these concepts.

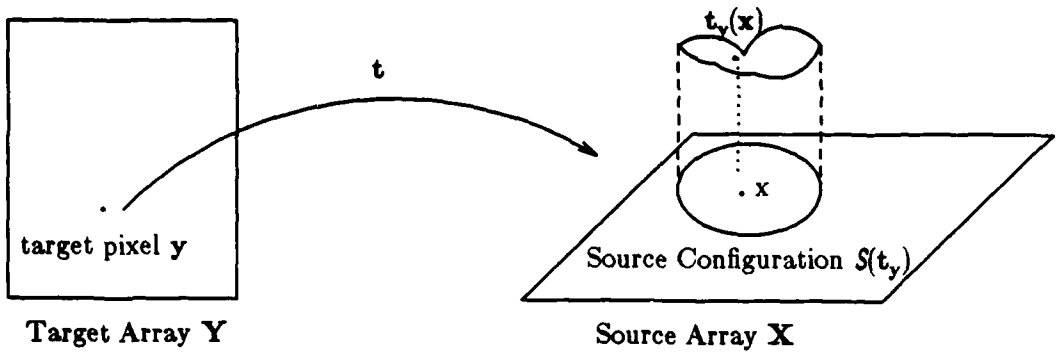


Figure 2. Pictorial Example of a Template from $\mathbf{Y}$ to $\mathbf{X}$

If $\mathbf{t} \in (\mathbf{R}^{\mathbf{X}})^{\mathbf{X}}$ or $\mathbf{t} \in (\mathbf{R}_{\infty}^{\mathbf{X}})^{\mathbf{X}}$, then $\mathbf{t}$ is called translation invariant if and only if for each triple $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z} \in \mathbf{X}$ with $\mathbf{y}+\mathbf{z}$ and $\mathbf{x}+\mathbf{z} \in \mathbf{X}$, we have that $t_y(\mathbf{x}) = t_{y+z}(\mathbf{x}+\mathbf{z})$. A template which is not translation invariant is called a translation variant or, simply, a variant template. A large class of translation invariant templates with finite support have the nice property that they can be defined pictorially. For example, let $\mathbf{X} = \mathbf{Z}^2$, where $\mathbf{Z}^2 = \mathbf{Z} \times \mathbf{Z} \subset \mathbf{R}^2$ is the set of discrete lattice points with integral coordinates. Let $\mathbf{y}=(x,y)$ be an arbitrary point of $\mathbf{X}$, $\mathbf{x}_1 = (x,y-1)$, $\mathbf{x}_2 = (x+1,y)$, and $\mathbf{x}_3 = (x+1,y-1)$. We now define a template $\mathbf{t} \in (\mathbf{R}^{\mathbf{X}})^{\mathbf{X}}$ by defining - for each $\mathbf{y} \in \mathbf{X}$ - its weights as $t_y(\mathbf{y}) = 1$, $t_y(\mathbf{x}_1) = 3$, $t_y(\mathbf{x}_2) = 2$, $t_y(\mathbf{x}_3) = 4$, and $t_y(\mathbf{x}) = 0$ if $\mathbf{x}$ is not an element of the set $\{\mathbf{y},\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3\}$. Note that it follows from our definition of $\mathbf{t}$ that $S(t_y) = \{\mathbf{y},\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3\}$. Thus $\mathbf{t}$ has configuration and weights as shown in Figure 3:

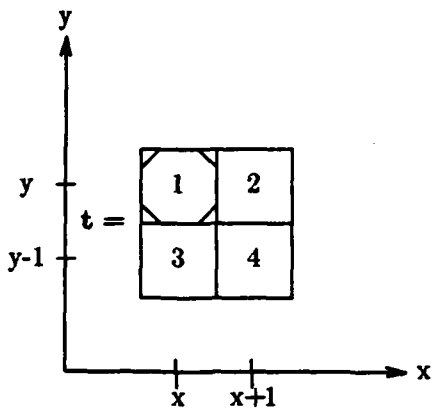


Figure 3. Pictorial Example of a Translation Invariant Template

The shaded cell in the pictorial representation of t indicates the location of the target point y.

Templates are used to define those image transformations which make use of all image values within some predescribed configuration of the source or input image. For example, given an image **a** on $\mathbf{X}$ and **t** a template from $\mathbf{Y}$ to $\mathbf{X}$, where $\mathbf{Y}$ may be of an entirely different shape, size or dimension than $\mathbf{X}$, then an operation between **a** and **t** will transform **a** into an image **b** on $\mathbf{Y}$ where each new pixel value $\mathbf{b}(\mathbf{y})$ is computed in terms of some arithmetic and/or logic combination of the values $\mathbf{a}(\mathbf{x})$ and $\mathbf{t}_\mathbf{y}(\mathbf{x})$, where $\mathbf{x}$ ranges over $S(\mathbf{t}_\mathbf{y})$ or $S_\infty(\mathbf{t}_\mathbf{y})$. Initially it may be convenient to view templates as masks such as the edge masks used in the Sobel or Kirsch edge detection schemes. It is important to realize, however, that the notion of a template is not the same as that of a mask. The magnification template presented in this report and Fourier templates presented in References 4 and 5 are examples of templates which are not masks. In the former, the target pixels are not even members of their respective configurations, while in the latter the weights and configurations change as a function of the position of the target pixel.

5. OPERATIONS BETWEEN IMAGES AND TEMPLATES

There are three basic template operations that are used to transform a real valued image. They are denoted $\oplus$, $\boxtimes$, and $\bigtriangledown$, and called generalized convolution, additive

maximum, and multiplicative maximum, respectively. For complex valued images only one image-template operation is defined, namely $\oplus$.

The template operations compute a pixel value $\mathbf{c(y)}$ by performing the basic operation of addition or maximum on a weighted collection of pixel values $\mathbf{a(x)}$, with coordinates $\mathbf{x}$ in some subset of $\mathbf{X}$. Let $\mathbf{X} \subset \mathbf{R}^n$ be finite and $\mathbf{Y} \subset \mathbf{R}^m$. Suppose $\mathbf{a} \in \mathbf{R}^{\mathbf{X}}$ and $\mathbf{t} \in (\mathbf{R}^{\mathbf{X}})^{\mathbf{Y}}$, then we define

$$\mathbf{a} \oplus \mathbf{t} \equiv \{(\mathbf{y},\mathbf{c(y)}) : \mathbf{c(y)} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{a(x)} \cdot \mathbf{t_y(x)},\ \mathbf{y} \in \mathbf{Y}\}$$

and

$$\mathbf{a} \boxtimes \mathbf{t} \equiv \{(\mathbf{y},\mathbf{c(y)}) : \mathbf{c(y)} = \bigvee_{\mathbf{x} \in \mathbf{X}} \mathbf{a(x)} + \mathbf{t_y(x)},\ \mathbf{y} \in \mathbf{Y}\},$$

where $\bigvee_{\mathbf{x} \in \mathbf{X}} \mathbf{a(x)} + \mathbf{t_y(x)} = \max\{\, \mathbf{a(x)} + \mathbf{t_y(x)} : \mathbf{x} \in \mathbf{X} \,\}$.

Several important comments are now in order. First note that since $\mathbf{t_y(x)} = 0$ whenever $\mathbf{x} \notin \mathcal{S}(\mathbf{t_y})$, we have that $\sum_{\mathbf{x} \in \mathbf{X}} \mathbf{a(x)} \cdot \mathbf{t_y(x)} = \sum_{\mathbf{x} \in \mathcal{S}(\mathbf{t_y})} \mathbf{a(x)} \cdot \mathbf{t_y(x)}$. Here we use the convention $\sum_{\mathbf{x} \in \mathbf{X}} \mathbf{a(x)} \cdot \mathbf{t_y(x)} = 0$ whenever $\mathcal{S}(\mathbf{t_y}) = \emptyset$. Thus, the new pixel value $\mathbf{c(y)}$ depends only on the values of $\mathbf{a(x)}$ and $\mathbf{t_y(x)}$ for $\mathbf{x} \in \mathcal{S}(\mathbf{t_y})$.

Similarly, if $\mathbf{t} \in (\mathbf{R}_{\infty}^{\mathbf{X}})^{\mathbf{Y}}$ and $\mathbf{t_y(x)} = -\infty$ whenever $\mathbf{x} \notin \mathcal{S}_{\infty}(\mathbf{t_y})$, then $\bigvee_{\mathbf{x} \in \mathbf{X}} \mathbf{a(x)} + \mathbf{t_y(x)} = \bigvee_{\mathbf{x} \in \mathcal{S}_{\infty}(\mathbf{t_y})} \mathbf{a(x)} + \mathbf{t_y(x)}$. Here we define $\bigvee_{\mathbf{x} \in \mathcal{S}_{\infty}(\mathbf{t_y})} \mathbf{a(x)} + \mathbf{t_y(x)} = -\infty$ whenever $\mathcal{S}_{\infty}(\mathbf{t_y}) = \emptyset$. Hence, the action for computing the new pixel value $\mathbf{c(y)}$ generally takes place over the set $\mathcal{S}(\mathbf{t_y})$ or $\mathcal{S}_{\infty}(\mathbf{t_y})$.

It may be apparent by now that the sets $\mathcal{S}(\mathbf{t_y})$ and $\mathcal{S}_{\infty}(\mathbf{t_y})$ generalize the notion of a convolution window or mask and a morphological structuring element, respectively. The application examples provided in this paper should clarify the analogy.

The operation of multiplicative maximum is analogous to that of the additive maximum. In particular, for $\mathbf{a} \in \mathbf{R}^{\mathbf{X}}$ and $\mathbf{t} \in (\mathbf{R}_{\infty}^{\mathbf{X}})^{\mathbf{Y}}$ with $\mathbf{t_y(x)} = -\infty$ whenever $\mathbf{x} \notin \mathcal{S}_{\infty}(\mathbf{t_y})$, we define

$$\mathbf{a} \boxdot \mathbf{t} \equiv \{(\mathbf{y},\mathbf{c(y)}) : \mathbf{c(y)} = \bigvee_{\mathbf{x} \in \mathcal{S}_{\infty}(\mathbf{t_y})} \mathbf{a(x)} \cdot \mathbf{t_y(x)},\ \mathbf{y} \in \mathbf{Y}\}$$

where $\bigvee_{\mathbf{x} \in \mathcal{S}_{\infty}(\mathbf{t_y})} \mathbf{a(x)} \cdot \mathbf{t_y(x)} = -\infty$ whenever $\mathcal{S}_{\infty}(\mathbf{t_y}) = \emptyset$.

There is an easy generalization of the above defined operations between images and templates that proves useful when expressing algorithms in image algebra code. Suppose $\mathbf{X} \subset \mathbf{R}^n$, $\mathbf{W} \subset \mathbf{R}^n$, $\mathbf{Y} \subset \mathbf{R}^m$, $\mathbf{a} \in \mathbf{R}^X$, and $\mathbf{t} \in (\mathbf{R}^W)^Y$ with $S(t_y)$ finite for each $y \in Y$. Then we define

$$\mathbf{a} \oplus \mathbf{t} \equiv \{(y, c(y)) : c(y) = \sum_{x \in X \cap S(t_y)} a(x) \cdot t_y(x),\ y \in Y\},$$

where $c(y) = 0$ whenever $X \cap S(t_y) = \emptyset$.

The operations $\boxtimes$ and $\bigtriangledown$ are defined in a similar fashion. For example, for $\boxtimes$ we define the new pixel value $c(y) = \bigvee_{x \in X \cap S_\infty(t_y)} a(x) + t_y(x)$.

The complementary operations of multiplicative minimum and additive minimum are defined in terms of $\bigtriangledown$ and $\boxtimes$ as follows:

$$\mathbf{a} \bigtriangleup \mathbf{t} \equiv -(\mathbf{a} \bigtriangledown -\mathbf{t})$$

$$\mathbf{a} \boxdot \mathbf{t} \equiv -(-\mathbf{a} \boxtimes -\mathbf{t})$$

In the above definitions we assume that $S(t_y)$ and $S_\infty(t_y)$ are finite for each $y \in Y$. However, the definitions extend to continuous functions $a(x)$ and t_y on compact sets $S(t_y)$ and $S_\infty(t_y)$, with the exception that in the formulation of $\mathbf{a} \oplus \mathbf{t}$ the sum is replaced by an integral. That is,

$$c(y) = \int_{S(t_y)} a(x) \cdot t_y(x) dx$$

Thus, image algebra operations can be used for expressing both continuous and discrete image transformations.

As a final comment we note that while $\mathbf{a} \in \mathbf{R}^X$, $\mathbf{a} \oplus \mathbf{t}$, $\mathbf{a} \boxtimes \mathbf{t}$, and $\mathbf{a} \bigtriangledown \mathbf{t}$ are all elements of $\mathbf{R}^Y$ (or $\mathbf{R}_\infty^Y$). Thus, template operations may be used for changing the dimensionality or size and shape of images. In particular, in addition to the usual local or global convolutions - as occur in edge enhancement, local smoothing, morphological operations and Fourier like transformations - template operations also provide a tool for image rotation, zooming, image reduction, masked extraction, and matrix multiplication.

At first glance, these operations may appear somewhat mysterious. However, the examples provided below and those given in References 6,7,8, and 9 illustrate the inherent

simplicity and power of these operations.

Example 1. Local Averaging. Let $\mathbf{a}$ be an image on a rectangular array $\mathbf{X} \subset \mathbf{Z}^2$. Let $\mathbf{Y} = \mathbf{Z}^2$ and $\mathbf{t} \in (\mathbf{R}^{\mathbf{Y}})^{\mathbf{Y}}$ be the 3x3 neighborhood template defined as follows:

$$\mathbf{t} = \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Then $\frac{1}{9}\mathbf{a} \oplus \mathbf{t}$ represents the image obtained from $\mathbf{a}$ by local averaging since the new pixel value is given by $\mathbf{c}(\mathbf{y}) = \frac{1}{9} \sum_{\mathbf{x} \in S(\mathbf{t}_\mathbf{y}) \cap \mathbf{X}} \mathbf{a}(\mathbf{x})$.

As an important remark, we note that the image $\mathbf{a} \oplus \mathbf{t}$ is an image on all of $\mathbf{Z}^2$ with zero values outside of the array $\mathbf{X} \subset \mathbf{Z}^2$. Obviously, computers are not capable of storing images defined on infinite arrays. Furthermore, in practice one is only interested in the image $\frac{1}{9}(\mathbf{a} \oplus \mathbf{t})$ restricted to the array $\mathbf{X}$, that is $\frac{1}{9}(\mathbf{a} \oplus \mathbf{t})|_\mathbf{X}$, where $|_\mathbf{X}$ denotes the restriction to $\mathbf{X}$.

This problem could be solved as follows: Let $\mathbf{s} \in (\mathbf{R}^{\mathbf{X}})^{\mathbf{X}}$ be defined by $\mathbf{s}_\mathbf{y} \equiv (\mathbf{t}_\mathbf{y})|_\mathbf{X}$ for each $\mathbf{y} \in \mathbf{X}$, where $\mathbf{t}$ is the template defined in Example 1. Then $\frac{1}{9}\mathbf{a} \oplus \mathbf{s}$ provides the desired finite image, since $(\mathbf{a} \oplus \mathbf{t})|_\mathbf{X} = \mathbf{a} \oplus \mathbf{s}$. Thus, the question arises: "Why not simply define $\mathbf{t}$ as a template from $\mathbf{X}$ to $\mathbf{X}$ instead from $\mathbf{Z}^2$ to $\mathbf{Z}^2$?"

The rationale for defining the template as we did is that this template can be used for smoothing any 2-dimensional image independent of its array size $\mathbf{X}$. The reason for this is that when defining an image $\mathbf{b}$ in a program one is usually forced to declare its dimensions, i.e. the size of its underlying array $\mathbf{X}$. In particular, an image algebra "program" statement of form $\mathbf{b} = \mathbf{a} \oplus \mathbf{t}$ means to replace $\mathbf{b}$ pointwise by $\mathbf{a} \oplus \mathbf{t}$ so that the value of $\mathbf{b}$ at location $\mathbf{y}$ is the value of $\mathbf{a} \oplus \mathbf{t}$ at location $\mathbf{y}$. That is, the array on the left side of the equality sign induces a restriction on the right side image array. In short, we make the convention that

the image algebra equation $\mathbf{b} = (\mathbf{a} \oplus \mathbf{t})|_{\mathbf{X}}$, where $\mathbf{X}$ is the domain of $\mathbf{b}$, corresponds to the image algebra program statement $\mathbf{b} = \mathbf{a} \oplus \mathbf{t}$. Thus, a programmer is not faced with the task of redefining $\mathbf{t}$ for a different sized image, as would be the case if he had defined $\mathbf{t} \in (\mathbf{R}^{\mathbf{X}})^{\mathbf{X}}$ for a given $\mathbf{X}$. In fact, this is the way we have embedded image algebra into image algebra FORTRAN, Reference 10.

Of course, the program statement $\mathbf{b} = \frac{1}{9}(\mathbf{a} \oplus \mathbf{t})$ will produce a boundary effect. In particular, if $\mathbf{a}$ and $\mathbf{b}$ are $m \times n$ images with underlying coordinate set $\mathbf{X} = \{(i,j) : 1 \leq i \leq m,\ 1 \leq j \leq n\}$, then

$$\mathbf{b}(1,1) = \frac{1}{9}(\mathbf{a}(1,1) + \mathbf{a}(1,2) + \mathbf{a}(2,1) + \mathbf{a}(2,2))$$

which is not the average of four points. One may either ignore this boundary effect or use one of several schemes to prevent it. For instance, one may simply avoid the boundary pixels by defining an array $\mathbf{Y} = \{ (i,j) \in \mathbf{X} : i \notin \{1,m\}$, and $j \notin \{1,n\} \}$ and the template $\mathbf{t} \in (\mathbf{R}^{\mathbf{X}})^{\mathbf{Y}}$ with $\mathbf{t}_y$ having configuration and weights as shown in Example 1. Then $\frac{1}{9}(\mathbf{a} \oplus \mathbf{t})$ represents the desired $(m-2) \times (n-2)$ output image. Letting m and n be variables again allows the application of $\mathbf{t}$ to any size images.

Example 2. Sobel Edge Detection. Let $\mathbf{a}$ be an image on a rectangular array $\mathbf{X} \subset \mathbf{Z}^2$, and $\mathbf{s}$, $\mathbf{t}$ the templates shown below defined on all of $\mathbf{Z}^2$. The image algebra expression

$$\lceil (\mathbf{a} \oplus \mathbf{s})^2 + (\mathbf{a} \oplus \mathbf{t})^2 \rceil^{1/2},$$

where

$\mathbf{s} =$

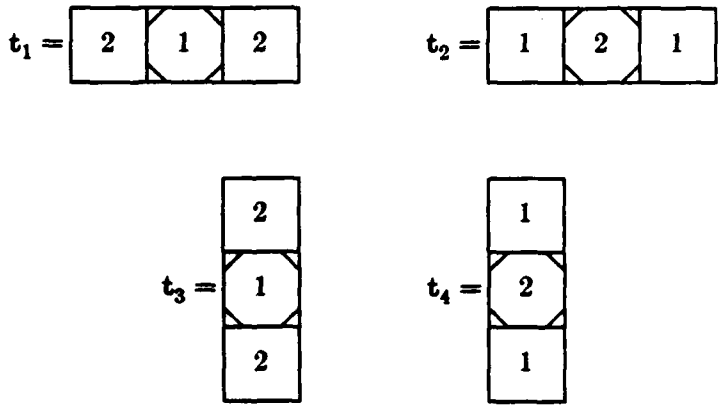
-1		1
-2		2
-1		1

$\mathbf{t} =$

-1	-2	-1
1	2	1

represents the Sobel edge enhanced image. The simplicity and translucency of image algebra expressions is apparent. Local averaging and Sobel edge detection expressed in the image algebra "look like" their corresponding textbook formulations, References 11 and 12. There is no lengthy code or obscure symbology involved in these expressions.

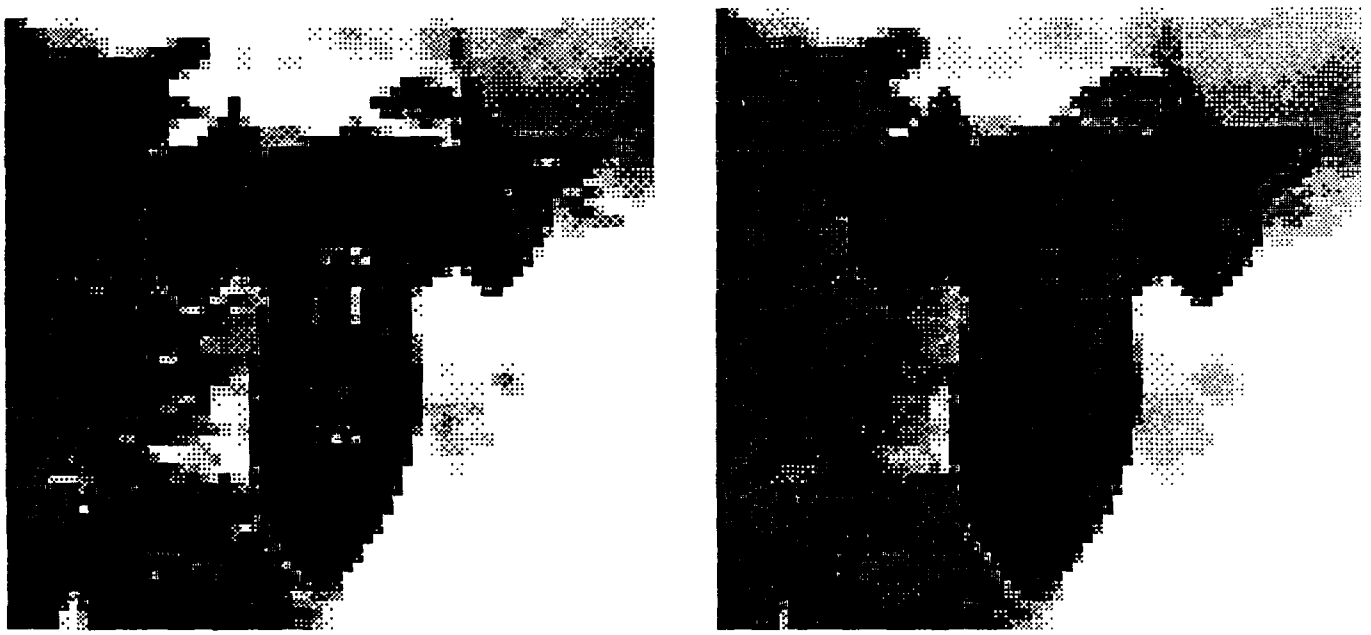
Example 3. Geometric Edge Filtering. This technique employs the operators $\textcircled{\vee}$ and $\textcircled{\wedge}$. Let $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{t}_3$, and $\mathbf{t}_4$ be the templates defined as follows:



Then the image algebra expression

$$\mathbf{b} = \{[\mathbf{a} \textcircled{\vee} \mathbf{t}_1 - \mathbf{a} \textcircled{\wedge} \mathbf{t}_2]^2 + [\mathbf{a} \textcircled{\vee} \mathbf{t}_3 - \mathbf{a} \textcircled{\wedge} \mathbf{t}_4]^2\}^{\frac{1}{2}}$$

is an edge enhancement technique which enhances edges while smoothing regions without sharp edge contrast. An example is shown in the figure below.



(a) (b)

Figure 4. (a) Input Image (b) Edge Enhanced Image

Example 4. Dilations and Erosions. We present this example for readers familiar with the basic notions of dilation, erosion and structuring elements that define all image processing schemes based on mathematical morphology. More details of the relationship between image algebra and mathematical morphology can be found in Reference 3.

The image algebra convolution operators $\boxtimes$ and $\boxtimes$ can be used to express the morphological operations of dilation and erosion, respectively, for both boolean and gray valued images. In particular, if B denotes the structuring element used in a dilation or an erosion, then we define a template $\mathbf{t}$ corresponding to B by setting $S_\infty(\mathbf{t}_y) = B_y'$, where B_y' denotes the reflection of B_y, and B_y the translation of B by the vector $\mathbf{y}$. The template weights $\mathbf{t}_y(\mathbf{x})$ are defined to be the values assigned to B_y' at location $\mathbf{x}$ and $\mathbf{t}_y(\mathbf{x}) = -\infty$ if $\mathbf{x} \notin B_y'$. Then

I $\mathbf{a} \boxtimes \mathbf{t}$ is equivalent to the dilation of $\mathbf{a}$ by B, and

II $\mathbf{a} \boxtimes -\mathbf{t}$ is equivalent to the erosion of $\mathbf{a}$ by B.

Statements I and II are true for both boolean and gray level dilation and erosions.

Figure 5(b) represents the dilation $\mathbf{a} \boxtimes \mathbf{t}$ of the boolean image $\mathbf{a}$ shown in Figure 5(a), while Figure 5(c) represents the erosion of the image $\mathbf{a} \boxtimes \mathbf{t}$, namely, $(\mathbf{a} \boxtimes \mathbf{t}) \boxtimes \mathbf{t}$ since in the boolean case $\mathbf{t}_y(\mathbf{x}) = 0$ if $\mathbf{x} \in S_\infty(\mathbf{t}_y)$ and, hence, $\mathbf{b} \boxtimes -\mathbf{t} = \mathbf{b} \boxtimes \mathbf{t}$. The template $\mathbf{t}$ used in this example is defined as follows:

$$\mathbf{t} = \begin{array}{|c|c|c|} \hline & 0 & \\ \hline 0 & 0 & 0 \\ \hline & 0 & \\ \hline \end{array}$$

(a) The Input Image $\mathbf{a}$ (b) The Dilated Image $\mathbf{b}=\mathbf{a} \boxtimes \mathbf{t}$ (c) The Eroded Image $\mathbf{b} \boxtimes \mathbf{t}$

Figure 5. Example of Dilation and Erosion

Note that the operation $\mathbf{a} \boxtimes \mathbf{t}$ simply replaces each pixel value of $\mathbf{a}$ by the maximum pixel value in the configuration of $\mathbf{t}$; i.e. each pixel value $\mathbf{a}(\mathbf{x})$ is compared with the values of its directly adjacent horizontal and vertical neighbors and the maximum value of these five possible pixel values replaces $\mathbf{a}(\mathbf{x})$. Similarly, for $(\mathbf{a} \boxtimes \mathbf{t}) \boxtimes \mathbf{t}$ each pixel value of $\mathbf{a} \boxtimes \mathbf{t}$ is replaced by the minimum pixel value in the configuration of $\mathbf{t}$.

It is important to note that a template may vary at different locations in both shape and weights. Thus the expression $\mathbf{a} \boxtimes \mathbf{t}$ may represent a far more complex algorithm than a simple dilation. In short, mathematical morphology, as used in actual image processing, is a

special substructure of the mathematical structure represented by the image algebra, Reference 3.

Before discussing parameterized templates, we introduce two special but very important image algebra operations. These are the sum and the maximum of an image. The sum of an image $\mathbf{a}$ on $\mathbf{X}$ is defined as

$$\sum \mathbf{a} \equiv \sum_{x \in X} \mathbf{a}(x)$$

and the maximum of $\mathbf{a}$ as

$$\bigvee \mathbf{a} \equiv \max\{\mathbf{a}(x) : x \in \mathbf{X}\}.$$

Of course, the sum and maximum can be expressed in terms of more elementary operations. Namely,

$$\sum \mathbf{a} \equiv \mathbf{a} \bullet \mathbf{1} \text{ and } \bigvee \mathbf{a} \equiv \sum(\mathbf{a} \textcircled{\scriptsize /} \mathbf{t}),$$

where $\mathbf{t}$ is a template from $\mathbf{Y} = \{0\}$ to $\mathbf{X}$ defined by $t_0(x) = 1$ for each $x \in \mathbf{X}$. Note that $\sum \mathbf{a} = \sum_{x \in X} \mathbf{a}(x)$ is a real number, $\mathbf{a} \textcircled{\scriptsize /} \mathbf{t}$ is an image consisting of a single point, and $\sum \mathbf{a} \textcircled{\scriptsize /} \mathbf{t}$ is the pixel value of the single point image.

Particularly nice examples that exhibit the brevity and translucency of image algebra code and involves image summation are order statistics of an image.

Example 5. Moments as Descriptors of Regions, Reference 13. For any image $\mathbf{a} = \{(x, \mathbf{a}(x)) : x \in \mathbf{X}\}$, with $\mathbf{X} \subset \mathbf{Z}^2$, moments of order pq are defined as follows:

$$m_{pq} = \sum_i \sum_j i^p j^q \mathbf{a}(i,j)$$

and central moments as $\mu_{pq} = \sum_i \sum_j (i-\bar{i})^p (j-\bar{j})^q \mathbf{a}(i,j)$ where $\bar{i} = \dfrac{m_{10}}{m_{00}}$, $\bar{j} = \dfrac{m_{01}}{m_{00}}$.

The image algebra translation of the moments is simply

$$m_{pq} = \sum(\mathbf{i}^p * \mathbf{j}^q * \mathbf{a})$$

where $\mathbf{i} = \{(x,y,i(x,y)) : i(x,y) = x\ ,\ (x,y) \in \mathbf{X}\}$ and $\mathbf{j} = \{(x,y,j(x,y)) : j(x,y) = y\ ,\ (x,y) \in \mathbf{X}\}$. Thus

$$m_{00} = \sum \mathbf{a}, \quad \bar{i} = \frac{\sum(\mathbf{i} * \mathbf{a})}{\sum \mathbf{a}} \text{ , and } \quad \bar{j} = \frac{\sum(\mathbf{j} * \mathbf{a})}{\sum \mathbf{a}}.$$

Defining the mean images $\bar{\mathbf{i}}$ and $\bar{\mathbf{j}}$ by

$$\bar{\mathbf{i}} = \frac{\mathbf{i} * \mathbf{a}}{\sum \mathbf{a}} \text{ and } \bar{\mathbf{j}} = \frac{\mathbf{j} * \mathbf{a}}{\sum \mathbf{a}}$$

the nonzero central moments are then given by the following translation:

$$\mu_{pq} = \sum [(\mathbf{i} - \bar{\mathbf{i}})^p * (\mathbf{j} - \bar{\mathbf{j}})^q * \mathbf{a}]$$

6. PARAMETERIZED TEMPLATES

Let $\mathbf{X}$, $\mathbf{Y}$ be coordinate sets and P a non-empty set. A parameterized $\mathbf{F}$ valued template from $\mathbf{Y}$ to $\mathbf{X}$ with parameters in P is a function of form

$$\mathbf{t}: P \rightarrow (\mathbf{F}^{\mathbf{X}})^{\mathbf{Y}}$$

Here we define $\mathbf{t}_p = \mathbf{t}(p)$. Thus for each $p \in P$, $\mathbf{t}_p$ is an $\mathbf{F}$ valued template from $\mathbf{Y}$ to $\mathbf{X}$. Again, in order to simplify notation, we define $\mathbf{t}_{p,y} \equiv (\mathbf{t}_p)_y$.

The set P is called the set of parameters and each $p \in P$ is called a parameter for $\mathbf{t}$.

Thus, a parameterized $\mathbf{F}$ valued template from $\mathbf{Y}$ to $\mathbf{X}$ gives rise to a family of regular $\mathbf{F}$ valued templates from $\mathbf{Y}$ to $\mathbf{X}$, namely $\{ \mathbf{t}_p : p \in P \}$. The following two examples should help clarify these notions.

Example 6. The Kirsch Edge Detector, References 14 and 11. The standard formulation of the Kirsch edge detection algorithm is to replace each pixel $\mathbf{a}(\mathbf{y})$ of the input image $\mathbf{a}$ by $\mathbf{c}(\mathbf{y}) = \max\{1, \max\{ | 5(a_i + a_{i+1} + a_{i+2}) - 3(a_{i+3} + ... + a_{i+7}) | : i = 1,2, ..., 8 \}\}$, where the addition of subscripts is mod 8 and the a_j's denote the following eight neighbors of $\mathbf{a}(\mathbf{y})$:

a_4	a_3	a_2
a_5	$\mathbf{a}(\mathbf{y})$	a_1
a_6	a_7	a_8

The image algebra expression representing Kirsch algorithm is given by

$$1 \vee (\overset{8}{\underset{i=1}{\vee}} | \mathbf{a} \oplus \mathbf{t}_i |),$$

where 1 denotes the unit image and $\mathbf{t}_i$ is defined as follows. For i = 1,2, ..., 8 define the parameterized template $\mathbf{t}_i$ by

$$\mathbf{t}_{i,y} = \{(\mathbf{x}, \mathbf{t}_{i,y}(\mathbf{x})) : \mathbf{t}_{i,y}(\mathbf{x}) = 5 \text{ if } \mathbf{x} = \mathbf{x}_i , \mathbf{x}_{i+1} , \mathbf{x}_{i+2} , \mathbf{t}_{i,y}(\mathbf{x}) = 3 \text{ if } \mathbf{x} = \mathbf{x}_{i+3} , ..., \mathbf{x}_{i+7} , \text{ else } \mathbf{t}_{i,y}(\mathbf{x}) = 0\} ,$$

where the $\mathbf{x}_j$'s denote the elements of $\mathcal{S}(\mathbf{t}_{i,y})$ as shown:

$\mathcal{S}(\mathbf{t}_{i,y}) =$

$\mathbf{x}_4$	$\mathbf{x}_3$	$\mathbf{x}_2$
$\mathbf{x}_5$	y	$\mathbf{x}_1$
$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$

Example 7. Image Magnification. Suppose $\mathbf{X} \subset \mathbf{R}^2$ is an m × n array, $\mathbf{Y} = \mathbf{Z}^2$, P = { p : p=($\mathbf{x}_0$,k), where $\mathbf{x}_0 \in \mathbf{X}$, k a positive integer }, and **a** an image on **X**. Given a pair of real numbers r = (r_1,r_2), define [r] ≡ ([r_1],[r_2]), where [r_i] denotes truncation of r_i to the nearest integer. For each $\mathbf{y} \in \mathbf{Y}$ and p=($\mathbf{x}_0$,k), define $\mathbf{t}_{p,y}(\mathbf{x}) = 1$ if $\mathbf{x} = [(\mathbf{y} - \mathbf{x}_0)/k + \mathbf{x}_0]$, and $\mathbf{t}_{p,y}(\mathbf{x})$ = 0 otherwise. Then $\mathbf{b} = \mathbf{a} \oplus \mathbf{t}_p$ represents the magnification of **a** by the factor k about the point $\mathbf{x}_0$. Thus, once this parameterized template has been defined, all a potential user of this template needs to supply is the magnification factor k, the point about which to magnify the image, and - in order to retain all the information - declare **b** to be of at least dimension km × kn. This example also shows how a template transformation is capable of changing the size of an image. Furthermore, note that from the definition of the template it follows that $\mathcal{S}(\mathbf{t}_{p,y}) = \{ \mathbf{x} : \mathbf{x} = [(\mathbf{y} - \mathbf{x}_0)/k + \mathbf{x}_0] \}$. In Figure 6 below, the image on the right represents the magnification of the image on the left by a factor of 2.

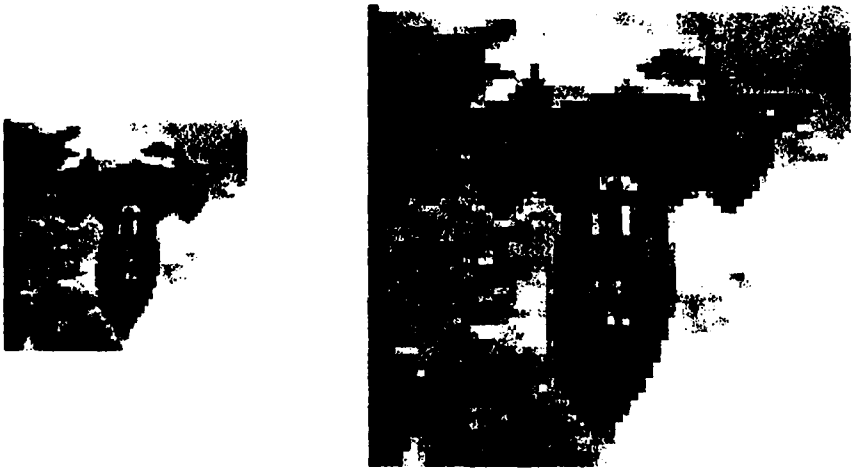


Figure 6. An Example of Image Magnification by a Factor of 2

7. OPERATIONS BETWEEN GENERALIZED TEMPLATES

The basic binary operations of addition, multiplication, and maximum between real valued images also constitute the basic binary operations between templates. In particular, if $\mathbf{s}$ and $\mathbf{t}$ are real valued templates from $\mathbf{Y}$ to $\mathbf{X}$, then addition, multiplication, and maximum between $\mathbf{s}$ and $\mathbf{t}$ are defined pointwise as follows:

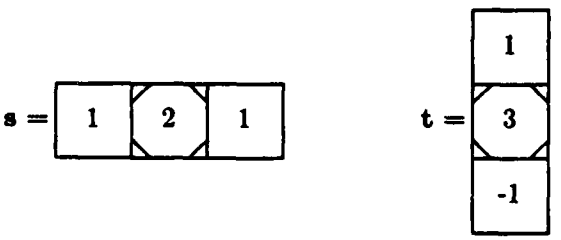
$$\mathbf{s} + \mathbf{t} \text{ by } (\mathbf{s} + \mathbf{t})_y = \mathbf{s}_y + \mathbf{t}_y$$

$$\mathbf{s} * \mathbf{t} \text{ by } (\mathbf{s} * \mathbf{t})_y = \mathbf{s}_y * \mathbf{t}_y$$

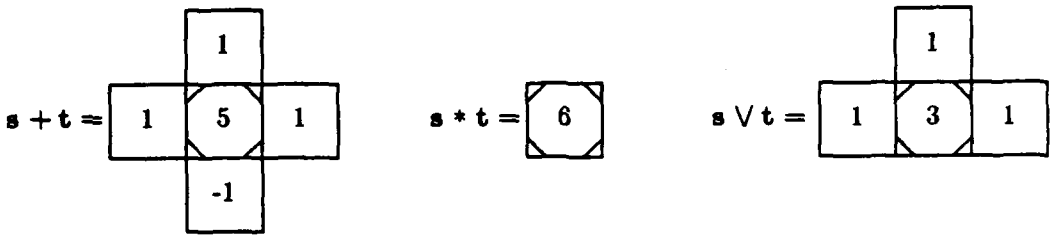
$$\text{and } \mathbf{s} \vee \mathbf{t} \text{ by } (\mathbf{s} \vee \mathbf{t})_y = \mathbf{s}_y \vee \mathbf{t}_y$$

Obviously, addition and multiplication are also defined for complex valued templates, while the maximum of two templates is defined for extended real valued templates as well.

Example 8. Template Arithmetic. Suppose $\mathbf{X} = \mathbf{Z}^2$ and $\mathbf{s} \in (\mathbf{R}^{\mathbf{X}})^{\mathbf{X}}$ and $\mathbf{t} \in (\mathbf{R}^{\mathbf{X}})^{\mathbf{X}}$ are the following translation invariant templates:



then



Subtraction, division, minimum, scalar multiplication, etc., can be defined from these basic operations in a straightforward manner. Thus, for example, $\mathbf{t}-\mathbf{s}$ is defined by $(\mathbf{t}-\mathbf{s})_y = \mathbf{t}_y - \mathbf{s}_y$.

The operations $\oplus$, $\boxtimes$, and $\otimes$ between images and templates generalize to operations between templates. In particular, if $\mathbf{t}$ is a real or complex valued template from $\mathbf{Y}$ to $\mathbf{X}$ and $\mathbf{s}$ is a real or complex valued template from $\mathbf{X}$ to $\mathbf{W}$, then we define the template $\mathbf{r} = \mathbf{s} \oplus \mathbf{t}$ from $\mathbf{Y}$ to $\mathbf{W}$ by defining the image function $\mathbf{r}_y$ by

$$\mathbf{r}_y(\mathbf{w}) = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{t}_y(\mathbf{x}) \cdot \mathbf{s}_x(\mathbf{w}), \quad \text{where } \mathbf{w} \in \mathbf{W}.$$

In order to compute the weights $\mathbf{r}_y(\mathbf{w})$ it is usually not necessary to sum over all of $\mathbf{X}$ but only a certain subset of $\mathbf{X}$. In particular, given $\mathbf{y}$, then for each $\mathbf{w} \in \mathbf{W}$ we define the set $S(\mathbf{w}) = \{\mathbf{x} \in \mathbf{X} : \mathbf{x} \in \mathcal{S}(\mathbf{t}_y) \text{ and } \mathbf{w} \in \mathcal{S}(\mathbf{s}_x)\}$. Then, since $\mathbf{t}_y(\mathbf{x}) \cdot \mathbf{s}_x(\mathbf{w}) = 0$ if $\mathbf{x} \notin S(\mathbf{w})$, we have that

$$\mathbf{r}_y(\mathbf{w}) = \sum_{\mathbf{x} \in S(\mathbf{w})} \mathbf{t}_y(\mathbf{x}) \cdot \mathbf{s}_x(\mathbf{w}),$$

where we define $\sum_{\mathbf{x} \in S(\mathbf{w})} \mathbf{t}_y(\mathbf{x}) \cdot \mathbf{s}_x(\mathbf{w}) = 0$ whenever $S(\mathbf{w}) = \emptyset$.

The operations $\mathbf{s} \boxtimes \mathbf{t}$ and $\mathbf{s} \otimes \mathbf{t}$ are defined in a similar fashion. Here we suppose that $\mathbf{t} \in (\mathbf{R}_\infty^{\mathbf{X}})^{\mathbf{Y}}$ and $\mathbf{s} \in (\mathbf{R}_\infty^{\mathbf{W}})^{\mathbf{X}}$ and define $S_\infty(\mathbf{w}) = \{\mathbf{x} \in \mathbf{X} : \mathbf{x} \in \mathcal{S}_\infty(\mathbf{t}_y) \text{ and } \mathbf{w} \in \mathcal{S}_\infty(\mathbf{s}_x)\}$. Then $\mathbf{r} = \mathbf{s} \boxtimes \mathbf{t} \in (\mathbf{R}_\infty^{\mathbf{W}})^{\mathbf{Y}}$ is defined by

$$\mathbf{r}_y(\mathbf{w}) = \bigvee_{\mathbf{x} \in S_\infty(\mathbf{w})} \mathbf{t}_y(\mathbf{x}) + \mathbf{s}_x(\mathbf{w}),$$

where we define $\bigvee_{\mathbf{x} \in S_\infty(\mathbf{w})} \mathbf{t}_y(\mathbf{x}) + \mathbf{s}_x(\mathbf{w}) = -\infty$ whenever $S_\infty(\mathbf{w}) = \emptyset$.

Similarly, $\mathbf{r} = \mathbf{s} \boxtimes \mathbf{t} \in (\mathbf{R}_{\infty}^{\mathbf{W}})^{\mathbf{Y}}$ is defined by

$$\mathbf{r}_{\mathbf{y}}(\mathbf{w}) = \bigvee_{\mathbf{x} \in S_{\infty}(\mathbf{w})} \mathbf{t}_{\mathbf{y}}(\mathbf{x}) \cdot \mathbf{s}_{\mathbf{x}}(\mathbf{w}),$$

where we define $\bigvee_{\mathbf{x} \in S_{\infty}(\mathbf{w})} \mathbf{t}_{\mathbf{y}}(\mathbf{x}) \cdot \mathbf{s}_{\mathbf{x}}(\mathbf{w}) = -\infty$ whenever $S_{\infty}(\mathbf{w}) = \emptyset$.

It follows from these definitions that $\mathcal{S}(\mathbf{r}_{\mathbf{y}}) = \{\mathbf{w} \in \mathbf{W} : S(\mathbf{w}) \neq \emptyset\}$, and $\mathcal{S}_{\infty}(\mathbf{r}_{\mathbf{y}}) = \{\mathbf{w} \in \mathbf{W} : S_{\infty}(\mathbf{w}) \neq \emptyset\}$.

The complementary operations $\boxtimes$ and $\circledwedge$ are defined by

$$\mathbf{s} \boxtimes \mathbf{t} = -(-\mathbf{s} \boxtimes -\mathbf{t})$$

and

$$\mathbf{s} \circledwedge \mathbf{t} = -(\mathbf{s} \circledvee -\mathbf{t}),$$

respectively.

Initially, these definitions seem to be fairly complex. The following examples serve to clarify these definitions and should provide a better understanding as to how composition of templates is accomplished.

Example 9. Template Convolution. Let $\mathbf{s}$ and $\mathbf{t}$ be the translation invariant templates defined in the previous example. Then

$\mathbf{s} \oplus \mathbf{t} =$

1	2	1
3	6	3
-1	-2	-1

and if $\mathbf{s}, \mathbf{t} \in (\mathbf{R}_{\infty}^{\mathbf{X}})^{\mathbf{X}}$, then

$\mathbf{s} \circledvee \mathbf{t} =$

1	2	1
3	6	3
-1	-2	-1

$\mathbf{s} \boxtimes \mathbf{t} =$

2	3	2
4	5	4
0	1	0

Template composition and decomposition are the primary reason for introducing operations between generalized templates. Composition and decomposition of templates provides a tool for algorithm optimization. For instance, if $\mathbf{s}$ and $\mathbf{t}$ are as in the example above and $\mathbf{r} = \mathbf{s} \oplus \mathbf{t}$, then computation of $\mathbf{a} \oplus \mathbf{r} = \mathbf{a} \oplus (\mathbf{s} \oplus \mathbf{t})$ by $(\mathbf{a} \oplus \mathbf{s}) \oplus \mathbf{t}$ uses 6 local multiplications instead of 9.

In general, if $\mathbf{r}$ is an nxn template, and $\mathbf{s}$ and $\mathbf{t}$ are decompositions of $\mathbf{r}$ into $1 \times n$ and $n \times 1$ templates, respectively, then the computation of $\mathbf{a} \oplus \mathbf{r}$ by $(\mathbf{a} \oplus \mathbf{s}) \oplus \mathbf{t}$ uses $2n$ multiplications instead of n^2. General methods for template decomposition and applications of decompositions to algorithm optimization can be found in Reference 7.

SECTION IV
MULTIVALUE OR MULTIDATA IMAGE ALGEBRA

1. HETEROGENEOUS ALGEBRAS AND HETEROGENEOUS VALUE SETS

Intuitively, an algebra is simply a collection of non-empty sets together with a finite number of operations (rules) for transforming one or more elements of the sets into another element of one of the sets. In this sense, the mathematics of image processing forms an algebra as it involves operations on and between elements of different sets such as operations on and between images, operations between images and templates, operations between templates, etc.

One of the main tools of algebra are isomorphisms. If two algebraic structures are isomorphic, that is, if there exists a one-to-one, operation preserving mapping of one onto the other, then they provide two different viewpoints of the same situation. The idea is that the more ways one has to look at a problem the better chance there is of solving it.

To make the notion of an algebra mathematically precise, we define an algebra as pair $(\mathcal{A},\mathcal{O})$ in which

1. $\mathcal{A} = \{\mathbf{A}_j\}$ is a family of non-empty sets of different types of elements and the subscripts j are members of some common indexing set J, and
2. $\mathcal{O} = \{o_i\}$ is a set of finitary operations, where each operation $o_k \in \mathcal{O}$ is a mapping

$$o_k : \prod_{i=1}^{n} \mathbf{A}_{i_k} \rightarrow \mathbf{A}_{m_k}$$

with each $\mathbf{A}_{j_k} \in \mathcal{A}$ and $\prod_{i=1}^{n} \mathbf{A}_{i_k}$ denotes the cartesian product

$$\prod_{i=1}^{n} \mathbf{A}_{i_k} = \mathbf{A}_{1_k} \times \mathbf{A}_{2_k} \times \cdots \times \mathbf{A}_{n_k} \equiv \{(a_{1_k}, \ldots, a_{n_k}) : a_{i_k} \in \mathbf{A}_{i_k}\}.$$

The operation o_k is unary if $n = 1$, binary if $n = 2$, ternary if $n = 3$, etc. The elements $\mathbf{A}_j$ of $\mathcal{A}$ are called the sets of operands of $\mathcal{A}$. Also, whenever the set of operations $\mathcal{O}$ is tacitly understood, it is customary to let $\mathcal{A}$ denote the algebra $(\mathcal{A},\mathcal{O})$.

Suppose $(\mathcal{A},\mathcal{O})$ and $(\mathcal{B},\mathcal{Q})$ are two algebras with $\mathcal{A} = \{\mathbf{A}_j\}$ and $\mathcal{B} = \{\mathbf{B}_i\}$. Then $(\mathcal{B},\mathcal{Q})$ is called a subalgebra of the algebra $(\mathcal{A},\mathcal{O})$ if $\mathcal{Q} \subset \mathcal{O}$ and for each $\mathbf{B}_i \in \mathcal{B}$ there exist $\mathbf{A}_j \in \mathcal{A}$ such that $\mathbf{B}_i \subset \mathbf{A}_j$.

An algebra which has only one set of operands is called a homogeneous algebra, while an algebra with more than one set of operands is called a heterogeneous algebra. For example, the real numbers $\mathbf{R}$ together with the operations of addition, multiplication, and maximum constitutes a homogeneous algebra that falls under the general class of algebras known as "lattice-ordered rings," Reference 15. On the other hand, the algebra $\mathcal{A}$ defined by $\mathcal{A} = \{\mathbf{R}, \mathbf{R}^{\mathbf{X}}, (\mathbf{R}^{\mathbf{X}})^{\mathbf{X}}\}$ and $\mathcal{O} = \{+, *, \bullet, \oplus\}$ constitutes a heterogeneous algebra which is a subalgebra of the image algebra defined in this paper. Various subalgebras of the image algebra are related in terms of isomorphisms to well-known algebraic structures such as linear algebra and mathematical morphology, References 7 and 3. It is these relationships of subalgebras of the image algebra to other well established mathematical structures or concisely defined areas of image processing which endow the image algebra with its versatility and power. All theorems, relationships, and "tricks-of-the-trade" associated with these structures can be interpreted and exploited within the language of the image algebra via these relationships. For example, in Reference 7 we demonstrated how several of these relationships can be applied to the problem of developing systematic techniques for the optimization and derivation of parallel algorithms. The main purpose of this section, however, is to extend the image algebra's capability to include operations for the manipulation of multivalued images.

Suppose $\mathbf{F} = \prod_{i=1}^{n} \mathbf{F}_i$, where each $\mathbf{F}_i$ is a value set whose set of finitary operations is $\mathcal{O}_i$. The set of naturally induced operations, $\mathcal{O}$, on $\mathbf{F}$ is defined as:

$$\mathcal{O} = \{ o \in \prod_{i=1}^{n} \mathcal{O}_i : \text{the coordinates of } o \text{ have the same arity } \}.$$

Thus, if $o = (o_1, \ldots\ldots, o_n)$ and o_i is a binary operation on $\mathbf{F}_i$ for some i between 1 and n, then $o \in \mathcal{O}$ only if each of the remaining coordinates o_j is a binary operation. The operation o is unary if its coordinates are unary operations, binary if its coordinates are binary operations, etc.

If $f = (f_1, \ldots\ldots, f_n) \in \mathbf{F}$ and $g = (g_1, \ldots\ldots, g_n) \in \mathbf{F}$ and $o = (o_1, \ldots\ldots, o_n) \in \mathcal{O}$ is binary, then we define

$$f \, o \, g \equiv (f_1 \, o_1 \, g_1, \ldots\ldots, f_n \, o_n \, g_n)$$

We now extend the notion of a value set to include sets of form $\mathbf{F} = \prod_{i=1}^{n} \mathbf{F}_i$ where each $\mathbf{F}_i$ is a

value set and the operations on $\mathbf{F}$ are the naturally induced operations. It follows that every value set is an algebra.

If $\mathbf{F} = \prod_{i=1}^{n} \mathbf{F}_i$ and $\mathbf{F}_i = \mathbf{F}_j$ for all i,j = 1,2,....,n, then $\mathbf{F}$ is called a homogeneous value set, otherwise $\mathbf{F}$ is called a heterogeneous value set. If $\mathbf{F}$ is homogeneous and $o = (o_1, \ldots\ldots, o_n)$ is an operation on $\mathbf{F}$ with $o_i = o_j$ for all i,j = 1,2,....,n, then o is called a homogeneous operation, otherwise o is called a heterogeneous operation. Whenever o is homogeneous, then it is customary to use o_i to denote the operation o. For instance, if $\mathbf{F} = \mathbf{F}_1 \times \mathbf{F}_2 \times \mathbf{F}_3$ and $\mathbf{F}_i = \mathbf{R}$ for i = 1,2, and 3, then $\mathbf{F} = \mathbf{R}^3$ and $\mathbf{F}$ is homogeneous. Furthermore, if $O_i = \{+, *, \vee\}$, then $+ = (+,+,+)$ and $\vee = (\vee, \vee, \vee)$ are homogeneous operations while $o = (+, *, \vee)$ is a heterogeneous operation. Applying these operations to the elements $a = (a_1, a_2, a_3)$ and $b = (b_1, b_2, b_3)$ of $\mathbf{R}^3$ yields

$$a + b = (a_1 + b_1, a_2 + b_2, a_3 + b_3), \quad a \vee b = (a_1 \vee b_1, a_2 \vee b_2, a_3 \vee b_3), \text{ and}$$

$$a \, o \, b = (a_1 + b_1, a_2 * b_2, a_3 \vee b_3).$$

2. OPERATIONS ON MULTIVALUED IMAGES

Let $\mathbf{a}$ be an $\mathbf{F}$ valued image on $\mathbf{X}$. If $\mathbf{F} = \prod_{i=1}^{n} \mathbf{F}_i$ and $n > 1$, then $\mathbf{a}$ is called a multivalued or multidata image. We distinguish between two types of multivalued images. If $\mathbf{a} \in \mathbf{F}^{\mathbf{X}}$ and $\mathbf{F}$ is heterogeneous, then $\mathbf{a}$ is called a heterogeneous multivalued image, and if $\mathbf{F}$ is homogeneous, then $\mathbf{a}$ is called a homogeneous multivalued image.

Given n sets $S_1, \cdots, S_n$, then the function $p_j : \prod_{i=1}^{n} S_i \rightarrow S_j$, where $1 \leq j \leq n$, defined by $p_j(s_1, \cdots, s_j, \cdots, s_n) = s_j$, is called the projection onto the jth coordinate or the jth coordinate projection. Projection functions play an important role in the manipulation of multivalued images. This is due to the fact that if $\mathbf{a} \in \mathbf{F}^{\mathbf{X}}$ is a multivalued image with $\mathbf{F} = \prod_{i=1}^{n} \mathbf{F}_i$, then - since $\mathbf{F}^{\mathbf{X}} = (\prod_{i=1}^{n} \mathbf{F}_i)^{\mathbf{X}}$ - the image $\mathbf{a}$ can always be viewed as a "stack" of n single-valued images $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n)$ where the ith coordinate image $\mathbf{a}_i$ of the stack is defined as

$$\mathbf{a}_i = p_i(\mathbf{a}) = \{(\mathbf{x}, \mathbf{a}_i(\mathbf{x})) : \mathbf{a}_i(\mathbf{x}) = p_i(\mathbf{a}(\mathbf{x}))\}.$$

Thus, $\mathbf{a}_i \in (\mathbf{F}_i)^{\mathbf{X}}$.

A typical LANDSAT image $\mathbf{a}$ of n-spectral bands provides a simple example of a multivalue image that can be viewed as a stack of single valued images. Here $\mathbf{F} = \prod_{i=1}^{n} \mathbf{F}_i = \mathbf{R}^n$ and $\mathbf{a}(\mathbf{x}) = (\mathbf{a}_1(\mathbf{x}),.....,\mathbf{a}_n(\mathbf{x}))$, with each $\mathbf{a}_i(\mathbf{x}) \in \mathbf{R}$.

The unary and binary operations on multivalued images are the operations induced by the set O of operations of the underlying value set $\mathbf{F}$. For instance, if $\mathbf{F} = \prod_{i=1}^{n} \mathbf{F}_i$, $\mathbf{a} = \{(\mathbf{x},\mathbf{a}(\mathbf{x})) : \mathbf{a}(\mathbf{x}) = (\mathbf{a}_1(\mathbf{x}),.....,\mathbf{a}_n(\mathbf{x}))\}$ and $\mathbf{b} = \{(\mathbf{x},\mathbf{b}(\mathbf{x})) : \mathbf{b}(\mathbf{x}) = (\mathbf{b}_1(\mathbf{x}),.....,\mathbf{b}_n(\mathbf{x}))\}$ are $\mathbf{F}$ valued images on $\mathbf{X}$, and $o = (o_1,.....,o_n) \in O$ a binary operation on $\mathbf{F}$, then

$$\mathbf{a}\, o\, \mathbf{b} \equiv \{(\mathbf{x},\mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = (\mathbf{a}_1(\mathbf{x})\, o_1\, \mathbf{b}_1(\mathbf{x}),.....,\mathbf{a}_n(\mathbf{x})\, o_n\, \mathbf{b}_n(\mathbf{x}))\}$$

or, equivalently,

$$\mathbf{a}\, o\, \mathbf{b} \equiv (\mathbf{a}_1\, o_1\, \mathbf{b}_1,.....,\mathbf{a}_n\, o_n\, \mathbf{b}_n).$$

If o is unary, then

$$o(\mathbf{a}) \equiv \{(\mathbf{x},\mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = (o_1(\mathbf{a}_1(\mathbf{x})),.....,o_n(\mathbf{a}_n(\mathbf{x})))\}.$$

or, equivalently,

$$o(\mathbf{a}) \equiv (o_1(\mathbf{a}_1),.....,o_n(\mathbf{a}_n)).$$

For example, if $\mathbf{F} = \mathbf{R}^n$ and o is a binary homogeneous operation with $o_i = +$, then

$$\mathbf{a} + \mathbf{b} = (\mathbf{a}_1 + \mathbf{b}_1,.....,\mathbf{a}_n + \mathbf{b}_n) = \{(\mathbf{x},\mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = (\mathbf{a}_1(\mathbf{x}) + \mathbf{b}_1(\mathbf{x}),.....,\mathbf{a}_n(\mathbf{x}) + \mathbf{b}_n(\mathbf{x}))\}$$

Similarly, if o is a unary homogeneous operation with $o_i = \sin$, then

$$\sin(\mathbf{a}) = (\sin(\mathbf{a}_1),.....,\sin(\mathbf{a}_n)) = \{(\mathbf{x},\mathbf{c}(\mathbf{x}) : \mathbf{c}(\mathbf{x})) = (\sin(\mathbf{a}_1(\mathbf{x})),.....,\sin(\mathbf{a}_n(\mathbf{x})))\}.$$

and if $o_i = \chi_{\geq}$, then

$$\chi_{\geq \mathbf{b}}(\mathbf{a}) = \{(\mathbf{x},\mathbf{c}(\mathbf{x})) : \mathbf{c}_i(\mathbf{x}) = 1 \text{ if } \mathbf{a}_i(\mathbf{x}) \geq \mathbf{b}_i(\mathbf{x}), \text{ otherwise } \mathbf{c}_i(\mathbf{x}) = 0,\ \mathbf{x} \in \mathbf{X}\}$$

It readily follows from these examples that if $\mathbf{F} = \mathbf{R}^n$, then the operations defined for $\mathbf{R}$ valued images extend in a natural way to $\mathbf{R}^n$ valued (vector valued) images. For example, if $\mathbf{a}$ and $\mathbf{b}$ are $\mathbf{R}^n$ valued images on $\mathbf{X}$, then the dot product is defined as

$$\mathbf{a} \bullet \mathbf{b} = \sum_{x \in X} \mathbf{a}(x)\mathbf{b}^t(x)$$

where $\mathbf{b}^t(\mathbf{x})$ denotes the transpose of $\mathbf{b}(\mathbf{x})$. The notion of scalar multiplication and addition are extended to vector multiplication and vector addition in a similar fashion. In particular, if $\mathbf{b} = \{(\mathbf{x},\mathbf{b}(\mathbf{x})) : \mathbf{b}_i(\mathbf{x}) = k_i \ \forall \mathbf{x} \in \mathbf{X}\}$, and $\mathbf{k}$ is the vector $\mathbf{k} = (k_1, \ldots, k_n) \in \mathbf{R}^n$, then we define

$$\mathbf{ka} \equiv \mathbf{b}*\mathbf{a} = \{(\mathbf{x},\mathbf{c}(\mathbf{x})) : \mathbf{c}_i(\mathbf{x}) = k_i\mathbf{a}_i(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}.$$

and

$$\mathbf{k} + \mathbf{a} \equiv \mathbf{b}+\mathbf{a} = \{(\mathbf{x},\mathbf{c}(\mathbf{x})) : \mathbf{c}_i(\mathbf{x}) = k_i+\mathbf{a}_i(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}$$

In the particular case where $\mathbf{b}$ is the constant vector valued image $\mathbf{b} = \{(\mathbf{x},\mathbf{b}(\mathbf{x})) : \mathbf{b}_i(\mathbf{x}) = k \ \forall \mathbf{x} \in \mathbf{X}$ and $1 \leq i \leq n\}$, we have that $\mathbf{k} = (k,\ldots,k)$, and we define

$$\mathbf{ka} \equiv \mathbf{ka} \equiv \mathbf{b}*\mathbf{a} \quad \text{and}$$

$$k + \mathbf{a} \equiv \mathbf{k} + \mathbf{a} \equiv \mathbf{b} + \mathbf{a}.$$

Obviously, these notions extend to $\mathbf{F} = \mathbf{C}^n$ valued images and vector valued images in general. In comparison to the next set of operations, the operations just discussed are not "new" operations but, as in the case of single valued images, the naturally induced operations of $(\mathbf{F},\mathcal{O})$.

Let $\mathbf{F} = \prod_{i=1}^{n} \mathbf{F}_i$ and $\mathbf{F}' = \prod_{i=1}^{k} \mathbf{F}'_i$ be two value sets. If $n < k$, then any function $g : \mathbf{F} \to \mathbf{F}'$ is called a data splitting function. If $n > k$, then g is called a data fusion function. In either case, g is of the form $g = (g_1, g_2, \ldots, g_k)$ where $g_i = p_i g$. Thus, the function g: $\mathbf{R} \to \mathbf{R}^2$ is defined by $g(r) = (\cos r, \sin r)$ is a value splitting function, while the function f: $\mathbf{R}^2 \to \mathbf{R}$ defined by $f(r_1,r_2) = r_1+r_2$ provides an example of a data fusion function.

Two elementary fusion/splitting functions of prime importance are the (generalized) projection and injection functions. Let $\mathbf{F} = \prod_{i=1}^{n} \mathbf{F}_i$ and $\mathbf{F}' = \prod_{m=1}^{k} \mathbf{F}_{j_m}$, where $\mathbf{F}_{j_m}$ denotes the j_mth factor of $\mathbf{F}$. Then the projection $p_{j_1 j_2 \ldots j_k} : \mathbf{F} \to \mathbf{F}'$, which deletes $n-k$ components from $\prod_{i=1}^{n} \mathbf{F}_i$, is defined by

$$p_{j_1 j_2 \ldots j_k}(f_1, f_2, \ldots, f_n) = (f_{j_1}, f_{j_2}, \ldots, f_{j_k}), \text{ where } f_{j_m} \in \mathbf{F}_{j_m}.$$

For $\mathbf{F}$ a homogeneous value set we define the injection function

$$q : \mathbf{F}_k \rightarrow \mathbf{F}, \text{ by } q(r) = (r, r, ..., r)$$

The ith-coordinate injection $q_i : \mathbf{F} \rightarrow \mathbf{F}$ is defined as $q_i \equiv qp_i$. Thus q_i replaces all the coordinate values of a point by the value of the ith-coordinate, namely

$$q_i(r_1, r_2, \ldots, r_i, \ldots, r_n) = (r_i, r_i, \ldots, r_i)$$

The basic operations on single and multivalued images together with the injection and projection functions can be combined to form useful image processing operations of arbitrary complexity. Projection functions, for example, can be used to reduce a multivalued image of n stacks to a single valued image or a multivalued image of k stacks, where $k < n$. For instance, if $\mathbf{a} \in \mathbf{F}^X$ with $\mathbf{F} = \mathbf{R}^n$, then we define

$$\sum \mathbf{a} \equiv \sum_{i=1}^{n} p_i(\mathbf{a}) = \mathbf{a}_1 + \mathbf{a}_2 + ... + \mathbf{a}_n$$

and

$$\bigvee \mathbf{a} \equiv \bigvee_{i=1}^{n} p_i(\mathbf{a}) = \mathbf{a}_1 \vee \mathbf{a}_2 \vee .... \vee \mathbf{a}_n .$$

As another example of obtaining new operations by combining previously defined operations with projections and injections, consider the case where $\mathbf{F} = \prod_{i=1}^{n} \mathbf{F}_i$ and $\mathbf{F}_j = \mathbf{R}$. Then we define the jth-coordinate maximum and minimum of two $\mathbf{F}$ valued images as

$$\mathbf{a} \vee|_j \mathbf{b} \equiv \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \text{ if } \mathbf{a}_j(\mathbf{x}) \geq \mathbf{b}_j(\mathbf{x}), \text{ otherwise } \mathbf{c}(\mathbf{x}) = \mathbf{b}(\mathbf{x}) \}$$

and

$$\mathbf{a} \wedge|_j \mathbf{b} \equiv \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \text{ if } \mathbf{a}_j(\mathbf{x}) \leq \mathbf{b}_j(\mathbf{x}), \text{ otherwise } \mathbf{c}(\mathbf{x}) = \mathbf{b}(\mathbf{x}) \},$$

respectively. Using injections and projections, these operations can be derived from the previously defined operations. In particular,

$$\mathbf{a} \vee|_j \mathbf{b} = \mathbf{a} * q_j[\chi_{\geq \mathbf{b}}(\mathbf{a})] + \mathbf{b} * q_j[\overline{\chi_{\geq \mathbf{b}}(\mathbf{a})}]$$

and

$$\mathbf{a} \wedge|_j \mathbf{b} = \mathbf{a} * q_j[\chi_{\leq \mathbf{b}}(\mathbf{a})] + \mathbf{b} * q_j[\overline{\chi_{\leq \mathbf{b}}(\mathbf{a})}].$$

In order to illustrate the use of multivalued image operations, we present a typical application example.

Example 10. Directional Edge Detection. The output of this directional edge detection scheme is a two-valued "edge" image in which each pixel has an intensity value as well as one of eight possible directional values. In this particular scheme, a grey scale image is convolved with the following four 3x3 edge masks (templates), with each mask corresponding to two possible direction.

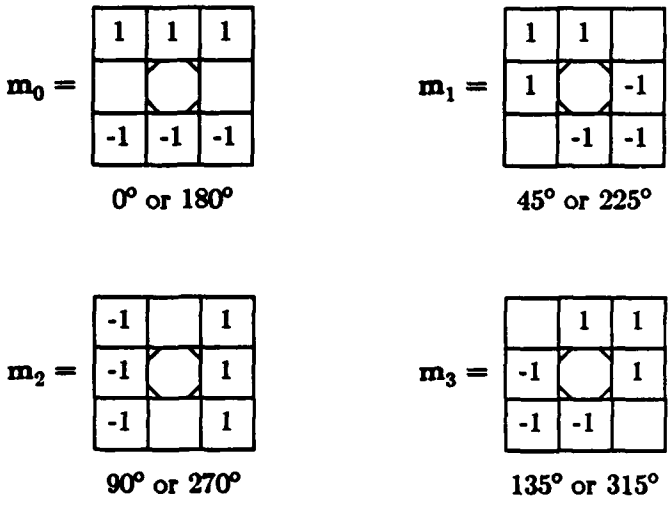


Figure 7. The Four Directional Edge Detection Masks

The resulting four images are then fused to form a single 2-valued image **b**. Data fusion is accomplished by assigning to the resultant pixel the value of the largest magnitude of the corresponding pixels in the four images and either assigning the direction θ associated with the mask of the convolved image if the specific pixel value of the convolved image is positive, or $\theta + 180°$ mod 360° if the value is negative. Thus, each pixel of **b** has both a magnitude and a direction associated with it. It is customary to use the integers 0 through 7 to represent the eight directions 0° through 315°, respectively. The addition "$\theta + 180°$ mod 360°" then becomes addition modulo eight, namely "i + 4 mod 8".

The image algebra translation of this algorithm is as follows. Let $f: \mathbf{R} \to \mathbf{R} \times \mathbf{Z}_8$ be defined by $f(r) = (|r|, 4\chi_{<0}(r))$, then

i. $\mathbf{a}_i = (0,i) + f(\mathbf{a} \oplus \mathbf{m}_i)$, i=0,...,4 and

ii. $\mathbf{b} = (V|_1)_{i=0}^{4} \mathbf{a}_i$

The next three figures provide a pictorial example of this algorithm.

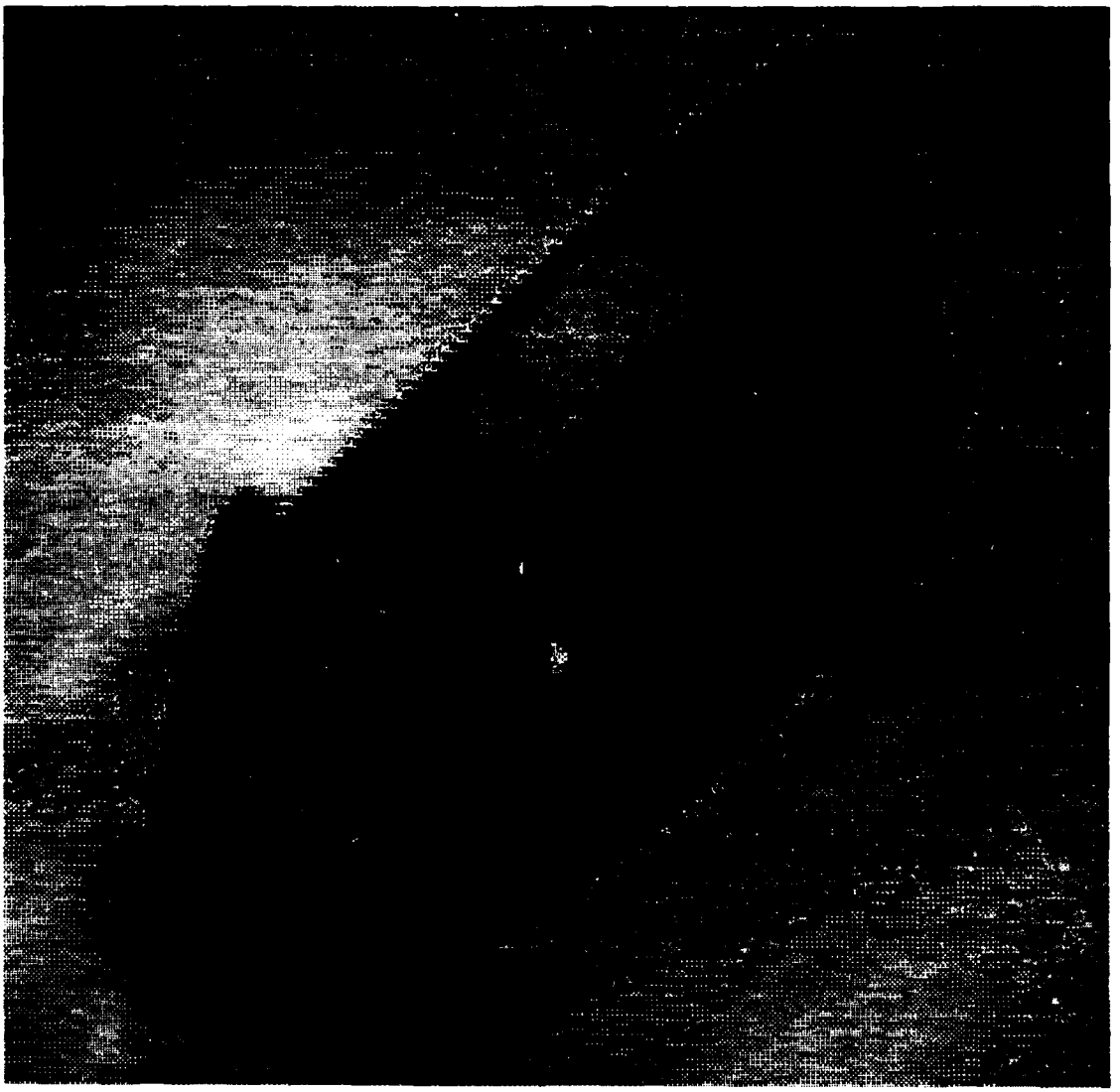


Figure 8. The Input Image **a**

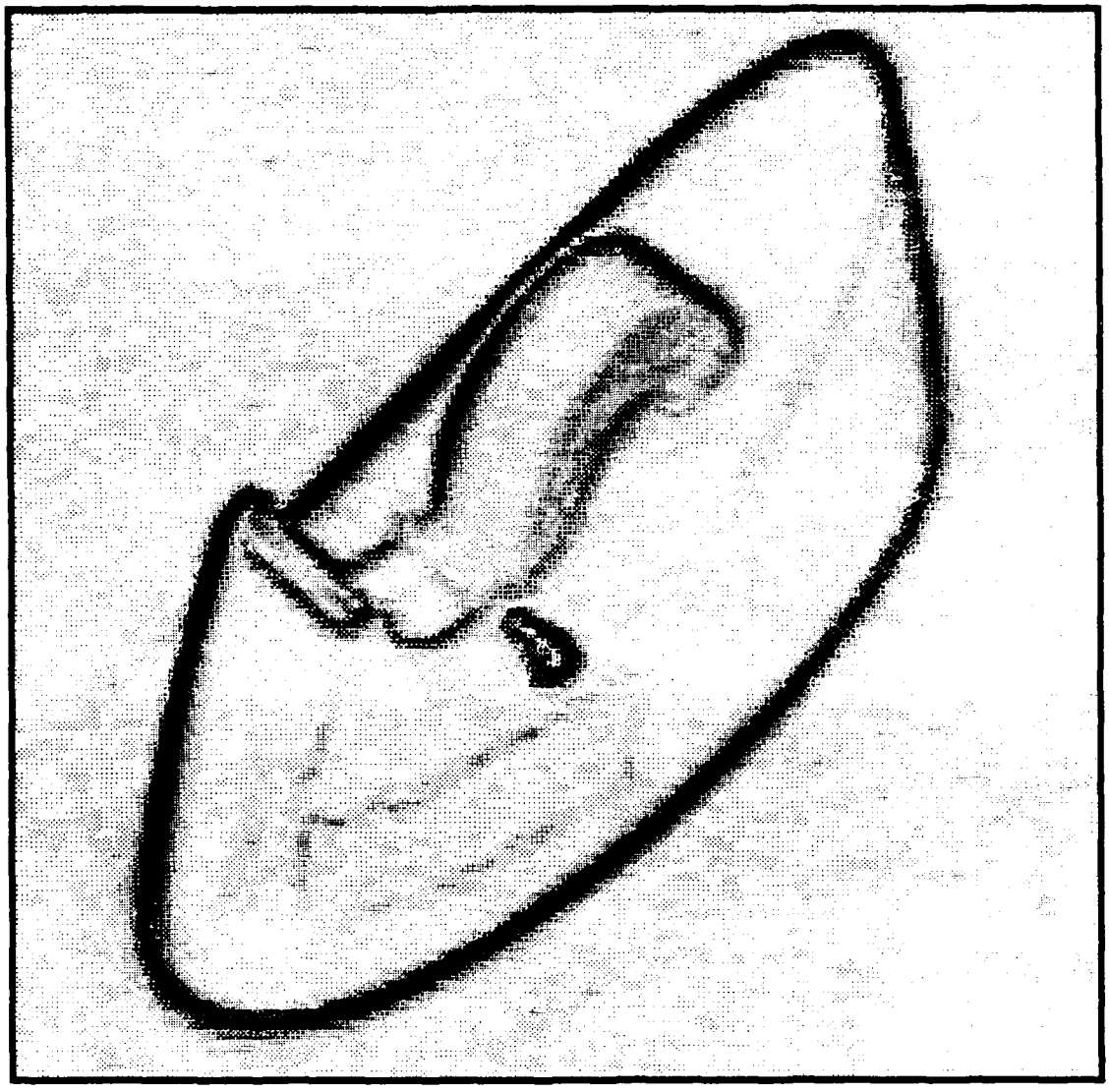


Figure 9. The Magnitude Image $p_1(\mathbf{b})$

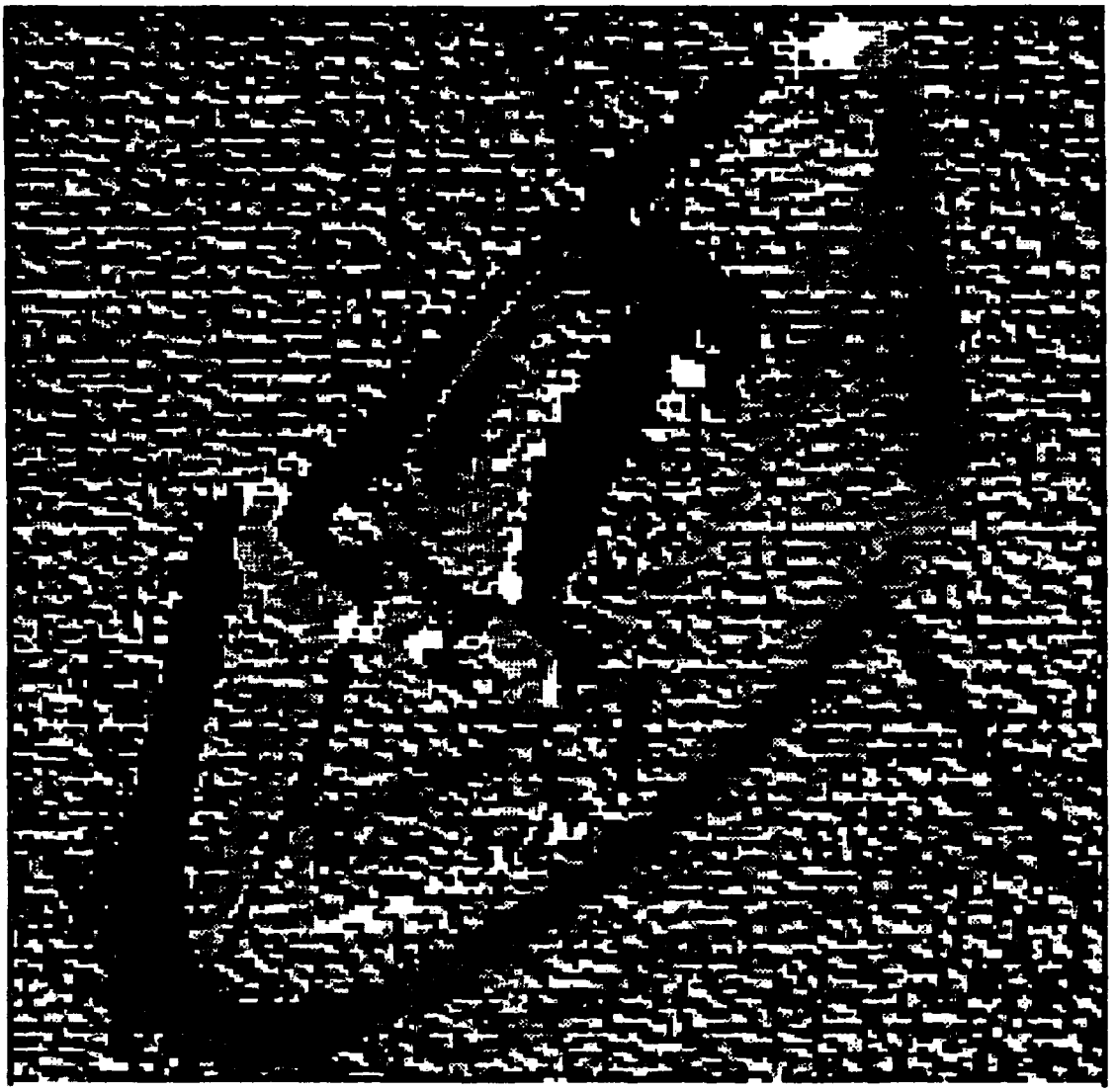


Figure 10. The Degree Image $p_2(\mathbf{b})$

3. MULTIVALUE TEMPLATES AND MULTIVALUE TEMPLATE OPERATIONS

If $\mathbf{F}=\prod_{i=1}^{n}\mathbf{F}_i$ and $n>1$, then $\mathbf{t}\in(\mathbf{F}^{\mathbf{X}})^{\mathbf{Y}}$ is called a multivalue or multilevel template from $\mathbf{Y}$ to $\mathbf{X}$. Analogous to multivalued images, a multilevel template $\mathbf{t}$ can be thought of as a stack of single valued templates $\mathbf{t}=(\mathbf{t}_1,\mathbf{t}_2,.....,\mathbf{t}_n)$, where each ith-coordinate $\mathbf{t}_i\in(\mathbf{F}_i^{\mathbf{X}})^{\mathbf{Y}}$ is

defined by $t_i \equiv p_i t$. Here p_i denotes the ith projection map $p_i : \mathbf{F} \rightarrow \mathbf{F}_i$ and the weights of t_{iy} are given by $p_i t_y(x)$.

Operations between multivalued images and multilevel templates are natural extensions of previously defined operations between single valued images and templates. Suppose $a \in \mathbf{F}^X$ and $t \in (\mathbf{F}^W)^Y$, where W and X are subsets of the same euclidean space. Let $o = (o_1, \ldots\ldots, o_n)$, where each o_i is an operation between $\mathbf{F}_i$ valued images on X and $\mathbf{F}_i$ valued templates from Y to W. Then $a\,o\,t$ is defined as

$$a\,o\,t \equiv (a_1\,o_1\,t_1, \ldots\ldots, a_n\,o_n\,t_n).$$

To illustrate this concept, consider the case where $\mathbf{F} = \mathbf{R}^3$ and $o = (\oplus, \boxtimes, \otimes)$. Then

$$a\,o\,t \equiv (a_1 \oplus t_1, a_2 \boxtimes t_2, a_3 \otimes t_3)$$

Also, if o is homogeneous, say $o = (\oplus, \oplus, \oplus)$, then - following our earlier convention - we set $o = \oplus$ and define

$$a \oplus t \equiv (a_1 \oplus t_1, a_2 \oplus t_2, a_3 \oplus t_3)$$

Thus, a multilevel template can have different configurations and can operate differently on the different levels of a multivalue image. Particular application examples of operations on multivalued images can be found in References 16 and 17.

As a final observation we note that if $\mathbf{F}$ and $\mathbf{F}'$ are two value sets with $\mathbf{F} \neq \mathbf{F}'$ and $o : \mathbf{F} \times \mathbf{F}' \rightarrow \mathbf{F}$ a binary operation, then the operations between $\mathbf{F}$ valued images and $\mathbf{F}$ valued templates can be generalized to operations between $\mathbf{F}$ valued images and $\mathbf{F}'$ valued templates. For example, if $\mathbf{F}$ is a vector space over the field of scalars $\mathbf{F}'$ (e.g., $\mathbf{F} = \mathbf{R}^n$ and $\mathbf{F}' = \mathbf{R}$), $a \in \mathbf{F}^X$ and $t \in (\mathbf{F}'^W)^Y$, where W and X are subsets of the same euclidean space, then we can define

$$c = a \oplus t \;\text{ by }\; c(y) = \sum_{x \in X \cap S(t_y)} t_y(x) \cdot a(x), \text{ where } y \in Y.$$

Thus c is a vector valued image with values in $\mathbf{F}$.

Obviously, if $\mathbf{F}$ is a vector lattice over $\mathbf{F}'$, then we can define $a \boxtimes b$ and $a \otimes b$ in a similar fashion. In the case where $a \in \mathbf{F}'^X$ and $t \in (\mathbf{F}^W)^Y$, the operation

$$\mathbf{c} = \mathbf{a} \oplus \mathbf{t} \text{ defined by } \mathbf{c}(\mathbf{y}) = \sum_{\mathbf{x} \in \mathbf{X} \cap S(\mathbf{t}_\mathbf{y})} \mathbf{a}(\mathbf{x}) \cdot \mathbf{t}_\mathbf{y}(\mathbf{x})$$

turns the scalar valued image **a** into a vector valued image **c**. Hence templates can be used to change not only the dimensionality, shape and size of an image, but also its value type.

SECTION V
DISCUSSION AND CONCLUSION

This document presents an analysis of the Singer-Kearfott algebra and an overview of the AFATL Standard Image Algebra. As we consider this paper an introduction to the subject and in order to keep the length of this paper within reasonable limits, we have purposely restricted application examples to a few simple and well-known image transforms. Proofs that the algebra is capable of expressing all image-to-image transformations have been presented elsewhere, References 6 and 9. In previous reports to AFATL we demonstrated the full power of the image algebra. In particular, we have shown how the image algebra encompasses such structures as linear algebra, polynomial algebra and the minimax algebra of economics and operations research, Reference 18, and how these relationships can be exploited for useful applications in image processing.

At first glance, it may seem that the image algebra lends itself well only to expressing parallel type of image processing operations. However, it turns out that the algebra is quite capable of expressing such sequential processes as chain encoding, Reference 19, in a more compact and translucent fashion than is possible within current higher level languages, Reference 20. It has been our experience that image processing algorithms requiring a variety of routines, including purely sequential ones, when coded in Image Algebra Fortran instead of Fortran have always resulted in significant code reduction, Reference 21.

In conclusion, the image algebra in its present form provides a comprehensive and unified algebraic structure for the representation of image to image operations. However, although image algebra operations on images can be used to extract statistical and geometric measures or image representations from images such as centroids, Euler number, and chain codes, we have made no serious attempts to extend the algebra to the symbolic domain. In particular, "high level" image operations which employ tools from such diverse areas as knowledge representation, graph theory, and surface representation have not been considered. Furthermore, the mathematics associated with the image algebra and its implications to image processing is, in itself, largely unchartered territory. Thus, the image algebra in its current state is not a finished product, but a continuously evolving mathematical theory concerned with the unification of image processing tasks.

REFERENCES

1. P.E. Miller, "Development of a Mathematical Structure for Image Processing," Optical Division Tech. Report, Perkin-Elmer (1983).

2. G. Birkhoff and J. Lipson, "Heterogeneous Algebras," *J. Combinatorial Theory* **8** (1970), 115-133.

3. G.X. Ritter, J.L. Davidson, and J.N. Wilson, "Beyond Mathematical Morphology," pp. 260-269 in *Proc. of SPIE Conf. - Visual Communication and Image Processing II*, Cambridge, MA (October 1987).

4. G.X. Ritter and J.N. Wilson, "The Image Algebra in a Nutshell," pp. 641-645 in *Proceedings of the First International Conference on Computer Vision*, IEEE Computer Society, London (June 1987).

5. P.D. Gader, "Image Algebra Techniques for Parallel Computation of Discrete Fourier Transforms and General Linear Transforms," Ph.D. Dissertation, University of Florida, Gainesville, FL (1986).

6. G.X. Ritter, M.A. Shrader-Frechette, and J.N. Wilson, "Image Algebra: A Rigorous and Translucent Way of Expressing All Image Processing Operations," in *Proc. of the 1987 SPIE Tech. Symp. Southeast on Optics, Elec.-Opt., and Sensors*, Orlando, FL (May 1987).

7. G.X. Ritter and P.D. Gader, "Image Algebra Techniques for Parallel Image Processing," *Journal of Parallel and Distributed Computing* **4**(5) (March 1987), 7-44.

8. G.X. Ritter and P.D. Gader, "Image Algebra Implementation on Cellular Array Computers," pp. 430-438 in *IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, Miami Beach, FL (1985).

9. G.X. Ritter and J.N. Wilson, "Image Algebra: A Unified Approach to Image Processing," in *Proceedings of the SPIE Medical Imaging Conference*, Newport Beach, CA (February 1987).

10. J.N. Wilson, D.C. Wilson, "Image Algebra Preprocessor," UF-CIS Technical Report, Dept. of Comp. and Info. Sci., Univ. of Florida, Gainesville, FL (1988).

11. W.K. Pratt, *Digital Image Processing*, John Wiley, New York (1978).

12. R.C. Gonzalez and P. Wintz, *Digital Image Processing,* Addison-Wesley, Reading, MA (1987).

13. M.K. Hu, "Visual Pattern Recognition by Moment Invariants," *IRE Transactions on Information Theory* **IT-8** (1962), 179-187.

14. R.A. Kirsch, "Computer Determination of the Constituent Structure of Biological Images," *Computers and Biomedical Research* **4**(3) (June 1971), 315-328.

15. Birkhoff, *Lattice Theory,* American Mathematical Society, Providence, RI (1984).

16. G.X. Ritter, J.L. Davidson, and J.N. Wilson, "Data Compression of Multispectral Images," in *Proceedings Soc. of Photo-Optical Instrumentation Engineers,* San Diego, CA (August, 1987).

17. G.X. Ritter, J.L. Davidson, and J.N. Wilson, "Image Algebra Application to Multisensor and Multidata Image Manipulation," in *Proc. of SPIE Tech. Symp. Optics, Elec.-Optics, and Sensors,* Orlando, FL (April 1988).

18. R. Cuninghame-Green, *Minimax Algebra: Lecture Notes in Economics and Mathematical Systems 166,* Springer-Verlag, New York (1979).

19. H. Freeman, "On the Encoding of Arbitrary Geometric Configurations," *IRE Trans. Electron. Comput.* **10** (1961).

20. G.X. Ritter, J.N. Wilson, and J.L. Davidson, "Image Algebra Application to Feature Measurement Extraction," UF-CIS Technical Report TR-88-04, Dept. of Comp. and Info. Sci., Univ. of Florida, Gainesville, FL (May 1988).

21. IVS Inc., "Image Algebra FORTRAN Version 2.0 Language Description and Implementation Notes," IVS-TR-88-02, Gainesville, FL (May 1988).